ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DOTTORATO DI RICERCA IN FISICA

CICLO XXXVI

Settore Concorsuale: 02/A2

Settore Scientifico Disciplinare: FIS/02

# Machine Learning for Quantum Systems and Quantum Optimization for Classical Problems


Presentata da

Simone Tibaldi

Supervisor

Prof.ssa Elisa Ercolessi

Co-supervisor

Dott. Edoardo Tignone

Coordinatore dottorato

Alessandro Gabrielli


Esame Finale Anno 2024

# Abstract

This manuscript is divided in two parts. In the first one we explore different machine learning inspired techniques to reconstruct the topological phase diagram of a few variants of the Kitaev chain and the extended Hubbard model in 1D. By applying four algorithms to a dataset of correlation functions we show how reliable this methods can be at predicting unknown phases and transfer the knowledge of one model to the other allowing also for interpretability of the results. In the second part we study Bayesian optimization and use it to run the Quantum Approximate Optimization Algorithm on the neutral atom quantum computer of the company PASQAL to solve a classical combinatorial problem. We show that the fast convergence and resilience to noise of the algorithm makes it suitable for this type of platforms. We conclude with an analysis of measurement-based quantum computation approach on neutral atoms with an estimation of resources needed to perform it.

# Contents

# Introduction

As suggested by the title, this thesis is divided in two distinct and yet symmetric parts that focus on two different types of problems that are hard to solve with two different types of approaches. On one side, we tackle the main problem lying at the heart of quantum many-body theory, which is the reconstruction of the phase diagram of a model, by using classical machine learning techniques. On the other side, we tackle a classical combinatorial optimization problem on a graph using quantum computation. The difficulty of the quantum problem of classifying the phases of a model is due to the fact that finding the ground-state of an interacting system is NP-hard[PW09]. The difficulty of the classical problem on a graph we considered is that it is strongly NP-hard [GJ78]. We collected in this work a series of results that can prove that classical techniques can improve research in quantum problems and viceversa.

This work is organized as follows: both parts are self-contained and therefore present an introduction (motivation) and a conclusion chapter. In the introductions there is a more detailed review of what is the problem being considered and motivations to why it has been the focus of this work. In the conclusions we recollect the objectives listed above and explain how and why they were respected.

The first part: *Classical for Quantum*, focuses on the reconstruction of the phase diagram of topological models with machine learning techniques. The reconstruction of the phase diagram is a central problem in fundamental physics because it allows you to predict the behaviour of a model and it has been approached with a variety of analytical and numerical techniques up to a few years ago. At the moment, we are seeing a rising number of approaches combining machine learning methods that have been proved extremely efficient at predicting the phase diagram. For this reason, in this part we present four machine learning techniques that were used especially for the job of classifying topological phases.

We clearly state the objectives of this part in the following:

1. provide re-usable techniques based on unsupervised learning algorithms that can be used to create a proposal of phase diagrams without assuming any a-priori knowledge of them;
2. prove that the information retained by this algorithms from the data of a model can be transferred to a more complicated related model;
3. give interpretation to the results obtained with machine learning being aware of the operations that are performed and their significance.

The second part: *Quantum for Classical*, focuses on quantum optimization on neutral atoms for combinatorial problems. This is part of a wide class of classical intractable problems that are being approached with the help of quantum computation. Since current quantum computers cannot yet run fault-tolerant deep algorithms, there is a lot of attention on hybrid quantum-classical algorithms – like the Quantum Approximate Optimization Algorithm (QAOA) – that alternate quantum circuits with classical optimization that can show quantum advantage at solving the classical task. These types of algorithms require a quantum variational (parametric) circuit and an efficient classical optimization routine. The project that inspired this work was in fact aimed at running QAOA on the quantum computer of the company PASQAL. The quantum part of the algorithm was therefore run on their neutral atoms quantum computer – a type of architecture that is based on trapped alkali atoms –in analogue mode, that is acting on all the qubits at once. For the classical part we studied and used Bayesian optimization that turned out to be ideal for the characteristic of the computer at PASQAL. During an internship at the company I also studied and expanded a quantum computation approach different from analogue, measurement-based quantum computation specifically suited for current state-of-the-art neutral atoms devices. Thus, I also present the results of this ongoing collaboration.

The objectives of this part are:

1. prove that Bayesian optimization is a reliable classical global optimizer suited for variational quantum algorithms like QAOA;
2. prove that we can efficiently solve QAOA on a neutral atoms quantum computer run in the analogue mode;
3. present the results of running our optimization scheme on the neutral atoms computer at PASQAL
4. present a new approach to measurement-based quantum computation on neutral atoms that I studied and expanded during an internship at the company PASQAL.

We end this general introduction by listing the main content of each chapter:

★ **Part I**

☐ Chapter 1. We introduce the topological models that were considered for the problem of identifying the topological phases. The models are: the Kitaev chain, the Kitaev chain with next-to-nearest neighbour (NNN) hopping, the interacting Kitaev chain, the Extended Hubbard model with a soft-shoulder potential.

☐ Chapter 2. Four unsupervised machine learning techniques to classify the phases of the models are presented: principal components analysis (PCA), $k$-means clustering, t-distributed stochastic neighbor embedding (t–SNE) and learning by confusion. Each algorithm is explained along with the methodology used for classifying the phases and then applied to each of the previous models. We show that the algorithms are efficient at classifying the topological phases, that knowledge can be transferred from one system to another and that we can predict also unknown phases of the Extended Hubbard model.

★ Part II

☐ Chapter 3. A detailed explanation of the combinatorial problem considered in this work and the Bayesian optimization algorithm is given. The algorithm is compared to other global optimizer to show better performance in term of calls to the quantum circuit and resilience to noise.

☐ Chapter 4. The neutral atoms platform of the company PASQAL that was used for the experiments is presented with explanation on how the QAOA algorithm was run in analogue mode. The chapter finishes showing the results of the experiment that prove that Bayesian optimization performed the task of optimization with success.

☐ Chapter 5. The chapter introduces the basics of measurement-based quantum computation both in its first conception and in the later-developed correlation approach. This work stems from the 6 months internship I spent at the company PASQAL in Paris. In the chapter it is also presented another approach developed for neutral atoms quantum computers that we expanded and used to estimate the resources needed to perform a standard quantum algorithm.

See the Declaration in the next page for further information on which papers this work was based on.

## Acknowledgements

This manuscript was produced thanks to the help of many colleagues and collaborators. First and foremost my supervisor Elisa Ercolessi, my co-supervisor Edoardo Tignone and D. Vodola for guiding me in the process of the PhD. My colleagues S. Pradhan, F. Dell'Anna, R. Cioli and C. M. Sanavio for very interesting and useful discussions. F. Caleca and G. Pupillo for the work on machine learning on the Hubbard model. The colleagues at CINECA, in particular D. Ottaviani and R. Mengoni. The colleagues at the PASQAL company: L. Leclerc for the help with accessing the quantum machine, L.-P. Henry and L. Vignoli for working with me during my 6 months spent at PASQAL preparing part of the work presented in this manuscript.

# Declaration

This manuscript is based on some published papers and some works that are still in production. In particular:

Chapter 2 is based on

- ✓ S. Tibaldi, G. Magnifico, D. Vodola, E. Ercolessi
  *Unsupervised and supervised learning of interacting topological phases from single-particle correlation functions*
  SciPost Phys. 14, 005 (2023)

- ▷ F. Caleca, S. Tibaldi, E. Ercolessi, G. Pupillo
  *Unsupervised learning of 1d Fermi-Hubbard model phase diagram*
  (In preparation)

Chapter 3 is based on:

- ✓ S. Tibaldi, D. Vodola, E. Tignone, E. Ercolessi
  *Bayesian Optimization for QAOA*
  IEEE Trans. on Quantum Engineering, 4:1–11 (2023)

Chapter 4 is based on:

- ▷ S. Tibaldi, L. Leclerc, E. Ercolessi
  *QAOA on neutral atoms quantum computer with Bayesian Optimization*
  (In preparation)

Chapter 5 is based on:

- ▷ S. Tibaldi, L. Vignoli
  *Measurement-based quantum computation on neutral atoms: resources estimation and improvement of the technique.*
  (In preparation)

A Work that is not included in the manuscript:

- ▷ C. M. Sanavio, S. Tibaldi, E. Tignone, E. Ercolessi
  *Quantum Circuit for Imputation of Missing Data*
  (In preparation)

# Part I

# Classical for Quantum

# Motivation: Machine Learning for Quantum Many-Body Systems

In the last few years research in physics has seen a new series of methods and practices inspired by and exploiting machine learning [MBW+19]. These new instruments have proved to be useful in applications in many-body quantum physics [CCC+19], in particular for finding a representation of a wavefunction [CT17] and describing its dynamics [CGG18], for reconstructing the wavefunction from experimental data [TMC+18], for speeding-up numerical simulations [CGG18], and for classifying quantum phases of matter of both synthetic [Wan16, vNLH17a, BCMT17, HSS17, SS20, RNS19] and experimental data [KDK+21, YZSD23]. A particular type of the latter is given by symmetry protected and topologically ordered phases, whose classification escapes from the standard Landau theory of spontaneous symmetry breaking [Sac11]. Indeed the combination of symmetries and topology can lead to new kinds of quantum phases that are characterized by a set of unusual features, such as *non-local* order parameters, the appearance of zero energy states at the boundary of the system, topological invariants, and long-range entanglement (for reviews see, for example: [Ber13, ZCZW19]). From an experimental point of view, topological materials have attracted much attention also because they represent a promising solution for physical implementation of qubits, more resilient to decoherence processes that affect devices based on superconducting or atomic physics technologies [Kit03, LP17].

Due to their intrinsically non-local features and the lack of local order parameters, the classification of topological phases is considered a very challenging task to tackle [CCC+19]. However, machine learning methods have been successfully applied to both non-interacting models where the topological invariant (winding number) representing each phase was known a priori [ZSZ18, SYZ+18], and to interacting models where the topological invariant cannot be obtained easily [HDW18].

The field of merging machine learning techniques with phase classifications of many-body models has seen increasing attention over the years. Starting from seminal works such as [Wan16] and [CM17] in which they showed that principal component analysis (the former) and simple neural networks (the latter) can identify the phase transition on the 2D Ising model, there has been great improvement in inventing new techniques, based on unsupervised learning as in [GVB+20], or completely original like the learning by confusion scheme [vNLH17b] that we exploited and extended in Section 2.4 of this work. At the moment, a lot of attention is given to models that exploit topological aspects of data in the classification like the diffusion

maps [KDK+21, RNS19, YD21, LYTS20, WKJC21].

Despite their success, in order to perform well, machine learning models require data sets with a very large number of training data [CHMT23], in the order of the thousands or millions. However, especially when handling interacting systems, it is not always easy to build such big data sets from both numerical simulations or experimental measurements.

This led us to the work presented in this chapter. One main goal is to study machine learning models trained on a data set obtained from a solvable system to be then applied to an interacting model which is obtained as an interacting generalization of the former, as in e.g. Ref. [MZvN+21]. In this way, insights about the features of a simple dataset can be exploited to characterize the phases of a more complicated one, saving simulation resources. Differently from what has been done in previous works [CGLN20, ZSZ18], our dataset will be constructed out of two-point single-particle correlation functions. These encode the properties of the different phases of the model and can be obtained numerically and measured experimentally [LG08, PMC+16, CNL+16], e.g. by using atom gas microscope with quenches to non-interacting models [GE21] or with randomized measurements [NEM+23].

The second goal of this work is to exploit unsupervised machine learning to explore unknown phases of models like the extended Hubbard model. We wish to show that it is possible to create a combined work-frame that exploits both machine learning predictions and analytical/numerical results from the theory of condensed matter. We care to do this in a reliable and reproducible way in order to give a first structure to a machine learning-assisted study of topological phases.

In this logic, we also acknowledge that despite the many works related to classification of topological or quantum phase transitions with machine learning and deep learning methods, there is often a lot of use of required *a priori knowledge* about the classification job. For this reason in this chapter we propose a series of techniques, both inspired and invented for the task of topological characterization of models, that try to be agnostic of the model considered and gives the idea of being universally applicable. To do so we compare our technique over a dataset that comprehends a total of 4 systems, either non–interacting and interacting data, with different data–size and data–types wishing to give a comprehensive analysis of the methods and how they can be used.

This first part of the thesis is organized in two chapters. The first chapter consists in an introduction of the models and the data produced for the analysis. In Sec. 1.1 we introduce the one dimensional Kitaev chain in its non-interacting [Kit01] and in Sec. 1.2 its interacting version [SASF11, HS12, TRS13, KST15, Mia17, FMV+20], in Sec. 1.3 the same model but with a next-to-nearest neighbour hopping term, Sec. 1.4 is for the extended Hubbard 1d model.

The second chapter lists the various methods used and the obtained results. We start with the most simple, yet effective, principal components analysis Sec. 2.1 and show that it is able to exploit the information from non-interacting systems and reproduce correct phase diagrams of their interacting version both for the Kitaev chain and its extended version. Then $k$-means in Sec. 2.2, a clustering algorithm

that also has the ability to transfer the information from a system to another and to predict many unknown phase transitions in the extended Hubbard model. Sec. 2.3 is for t-SNE another clustering algorithm and finally in Sec. 2.4 we exploit the famous learning by confusion method to corroborate the results of the previous sections. We finish this part of the manuscript with our conclusions in Sec. 2.4.3.

# 1 | Quantum Many-Body Models

In this chapter we introduce the quantum many-body models that interested this work. For each model we introduce the Hamiltonian and give an explanation of the phase diagram obtained by varying two Hamiltonian parameters. We give more details for the Kitaev chain which is more central to this work and just mention the relevant details for the other models. For each model we also explained the type of data that was generated and used in the following chapter to classify the phases with machine learning techniques. The models are:

- ▷ Sec. 1.1 Kitaev chain
- ▷ Sec. 1.2 Interacting Kitaev chain
- ▷ Sec. 1.3 Kitaev chain with next-to-nearest neighbour hopping (Kitaev NNN)
- ▷ Sec. 1.4 Extended Hubbard model

## 1.1 Kitaev Chain

In this section we introduce the Kitaev chain in its non–interacting version. We will present the phase diagram of the former and introduce the correlators that will be later used as a training set for the algorithms.

The one–dimensional Kitaev chain [Kit01] is a pedagogical model to show superconducting and topological effects. Given a chain of $L$ sites the Hamiltonian of the non–interacting (NI) version can be written as:

$$H^{\mathrm{NI}} = \sum_i \left( J a_i^\dagger a_{i+1} + \Delta\ a_i a_{i+1} + \mathrm{h.c.} \right) + \mu \sum_i a_i^\dagger a_i. \tag{1.1}$$

The operators $a_i(a_i^\dagger)$ create (annihilate) spinless fermions on site $i$, $J$ is the nearest neighbour hopping coefficient, $\Delta$ the superconducting pairing and $\mu$ the chemical potential.

We consider anti-periodic boundary conditions to diagonalize $H^{\mathrm{NI}}$ by going to momentum space using the standard Fourier transform (the choice of conditions is to avoid cancellation of terms like $\langle a_i a_j \rangle$ [VLE+14]). In this way, the Hamiltonian is translated into its Brillouin zone (BZ) version $H^{\mathrm{NI}} = \sum_{k \in BZ} H(k)$, where $H(k) = h_z(k)\sigma^z + h_y(k)\sigma^y$ is a two-band Hamiltonian, $k$ is the lattice quasi momentum and $\sigma^x$, $\sigma^y$ are Pauli matrices. In particular the expression for the single–particle

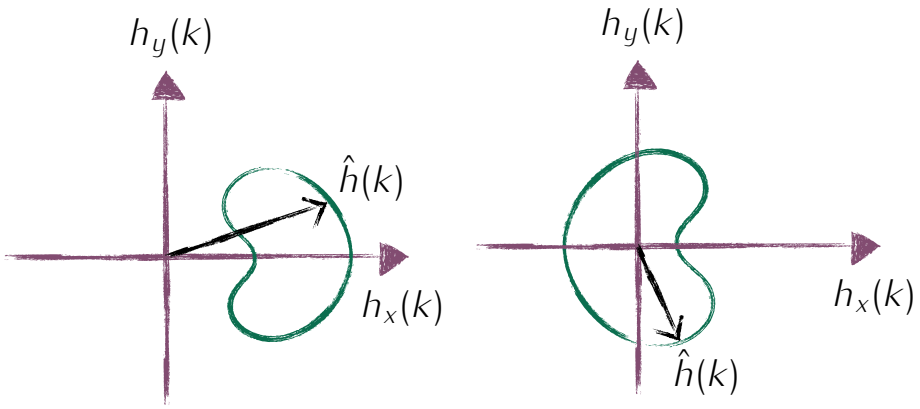



**Figure 1.1: Hamiltonian vector windings in the 1D**. The image shows an example of possible windings $\nu$ of the Hamiltonian vector $\hat{h}(k)$ when $k \in [0, 2\pi]$. (Left) $\nu = 0$. (Right) $\nu = 1$.

functions are:

$$h_z(k) = J\cos k + \mu/2, \quad h_y(k) = \Delta \sin k. \tag{1.2}$$

Now, we perform a Bogoliubov transformation that casts $H^{\text{NI}}$ in a diagonal form $H^{\text{NI}} = \sum_k E(k)\eta_k^\dagger \eta_k$, where $\eta_k$ are Bogoliubov operators and the single-particle energy $E(k)$ is given by

$$E(k) = \sqrt{h_z(k)^2 + h_y(k)^2}. \tag{1.3}$$

This model describes a one-dimensional topological superconductor belonging to the BDI symmetry class [SRFL08, CTSR16, KdBvW+17, SMJZ13]. This class has the particularity that the Hamiltonian vector $\hat{h}(k) = \vec{h}(k)/|\vec{h}(k)|$ with $\vec{h}(k) = (h_y(k), h_z(k))$ is a continuous map from the 1D BZ to the circle $S^1$. If we let the quasi-momentum $k$ slide from 0 to $2\pi$ (the 1D BZ) and count the number of times the vector $\hat{h}(k)$ turns around the origin we obtain the number winding $\nu$ (see Fig. 1.1). The winding number can be used to classify the model into different topological phases.

**Phase diagram** For our model, the winding number allows us to divide the Kitaev chain phase diagram into three phases. A trivial phase with winding number $\nu = 0$ and two superconducting topological phases with $\nu = \pm 1$. The phase diagram showing these phases obtained varying $(\mu, \Delta)$ keeping $J = 1$ is shown in Fig. 1.2(a). Notice that the phase diagram is symmetric for $\mu \leftrightarrow -\mu$.

The physical significance of these phases can be understood by transforming the fermionic operators into Majorana operators like so:

$$a_i = \frac{\gamma_{2i-1} + i\gamma_{2i}}{2}, \quad a_i^\dagger = \frac{\gamma_{2i-1} - i\gamma_{2i}}{2} \tag{1.4}$$

where $\gamma_{2i-1}$, $\gamma_{2i}$ are the Majorana operators respecting the anti-commutation relation $\{\gamma_i, \gamma_j\} = \delta_{ij}$. This relation entails that $\gamma_i^\dagger = \gamma_i$ meaning that creating a Majorana

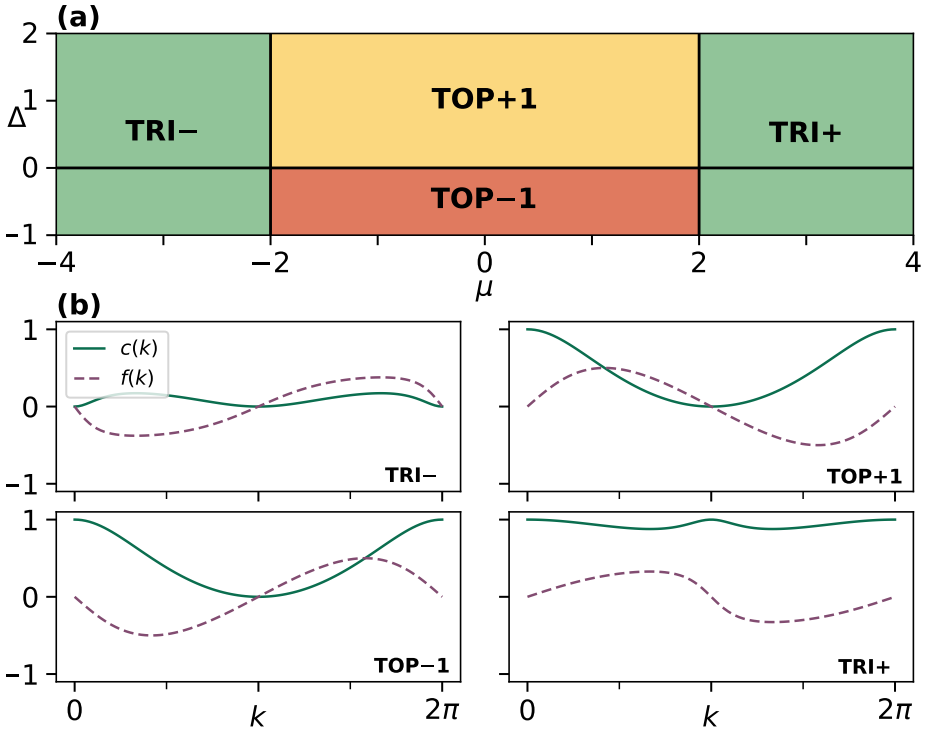



**Figure 1.2: Phase diagram of the Kitaev chain**. (a) The model presents two sectors: a trivial one for $|\mu| > 2$ which we divided into the phases TRI− and TRI+ and a topological sector, divided into two topological phases by the different winding numbers $\nu = \pm 1$ for the phases TOP+/TOP−1 respectively. (b) The correlation functions of this model for the different topological phases.

fermion is equal to destroying it and that $\gamma_i^2 = 1$ meaning that they do not respect the Pauli exclusion principle.

Rewriting Hamiltonian (1.1) using the Majorana operators we obtain:

$$H_\gamma = \frac{i}{2}\sum_j \Big[ -\mu(\gamma_{2j-1}\gamma_j) + (t + |\Delta|)\gamma_{2j}\gamma_{2j+1} + (-t + |\Delta|)\gamma_{2j-1}\gamma_{2j+2}\Big]. \tag{1.5}$$

We can consider two different cases:

- $|\Delta| = t = 0, \mu < 0$. This takes (1.5) into the form

$$H_\gamma = -\frac{\mu}{2}\sum_i \left(\gamma_{2i-1}\gamma_{2i}\right) \tag{1.6}$$

  where the Majorana operators $\gamma_{2i-1}$ and $\gamma_{2i}$ are paired inside the same site (upper part of Fig. 1.3). In this way, we obtain a groundstate with vanishing occupancy number which means that there are no Majorana fermions emerging. For this reason, this is called the *trivial phase* and it corresponds to the $\nu = 0$ phase.

- $|\Delta| = t > 0, \mu = 0$. Setting these parameters we obtain:

$$H_\gamma = it\sum_i^{N-1} \gamma_{2i}\gamma_{2i+1} \tag{1.7}$$

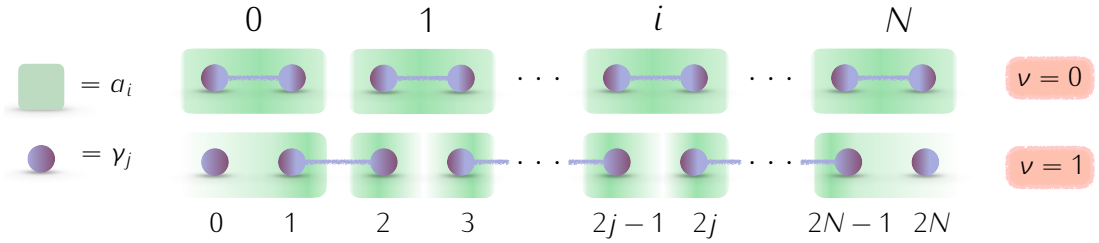



**Figure 1.3: Majorana Sites** Representation of the Majorana modes composing the sites of a topological model like the Kitaev chain. We can split each site (numerated $0, 1, \cdots N$) into two Majorana modes. In the trivial phase ($\nu = 0$) the modes of each pairs interacts more strongly between each other, in the topological phase ($\nu = 1$) the interaction is stronger between sites giving the freedom to the boundary modes to detach from the chain with zero energy cost.

the main difference with the previous phase is that now the right fermion of a site is paired to the left fermion of the next one (see lower part of Fig. 1.3). This corresponds to the *topological phase* that is characterized by a winding number $\nu = \pm 1$. This configuration suggests that there are two Majorana modes (at the far left/right of the chain) coupled between themselves but not to any other mode and thus have a null contribution to the energy. Consequently, we have a degeneracy of the groundstate due to the fact that we can either have zero fermions in the absence of the edge modes.

Therefore, $\nu$ corresponds to the number of (couples of) zero energy states that the model hosts at the boundaries of the chain when open boundary conditions are considered. This is a well–known fact in the literature as bulk-edge correspondence [BH13, AOP15]. For $\nu = 0$ no edge states will be present, while for $\nu = \pm 1$ an edge state on each boundary appears, as show in Figure 1.3.

**Correlators data** We define the Fourier transform of the single-particle standard ($c(k)$) and anomalous ($f(k)$) correlation functions:

$$c(k) = \sum_{i,j} e^{\mathrm{i}k(i-j)} \langle a_i^\dagger a_j \rangle \tag{1.8}$$

$$f(k) = \sum_{i,j} e^{\mathrm{i}k(i-j)} \langle a_i a_j \rangle \tag{1.9}$$

where the expectation values are taken over the ground state. Notice that $c(k)$ is real whereas $f(k)$ is purely imaginary. This is due to the antisymmetry of the expectation value $\langle a_i a_j \rangle$ for the exchange $i \leftrightarrow j$. For this reason we only take the imaginary part of $f(k)$. In our non–interacting model (Eq. (1.1)), the correlators $c(k)$ and $f(k)$ can be computed analytically and take the form:

$$c(k) = \frac{1}{2} + \frac{\mu/2 + J\cos k}{2E(k)}, \tag{1.10}$$

$$f(k) = \frac{\Delta \sin k}{2E(k)}, \tag{1.11}$$

with $E(k)$ being the energy dispersion relation 1.3. The typical form of one pair of correlators for each phase of the phase diagram is represented in panel (b) of Figure 1.2.

The correlators eqs. (1.8) and (1.9) are similar to the components of the Hamiltonian vector $\hat{h}(k)$ (1.2), in fact by a simple comparison we can check that:

$$c(k) = \frac{1}{2} + \frac{h_z(k)}{2E(k)}, \quad f(k) = \frac{h_y(k)}{2E(k)} \tag{1.12}$$

This is interesting because the winding number is calculated with $h_z(k)$, $h_y(k)$ [BH13]:

$$\nu = \int_0^{2\pi} \frac{h_y(k)\partial h_z(k) - h_z(k)\partial h_y(k)}{E(k)} \tag{1.13}$$

This makes us believe that the winding number can be extracted from $c(k)$ and $f(k)$. The correlation functions $c(k)$ and $f(k)$ will be used for building a non-interacting training set $S$ where each data point is labelled with the winding number of its corresponding phase. It is interesting to consider them as a dataset because they are typically measured in experiments to recover information about a system [LG08, PMC+16, CNL+16].

## 1.2 Interacting Kitaev Chain

For this model we add a nearest neighbor interaction term to the Hamiltonian (1.1) of the previous section to obtain:

$$H^{\mathrm{I}} = \sum_i (J a_i^\dagger a_{i+1} + \Delta\ a_i a_{i+1} + \text{h.c.}) + \mu \sum_i n_i + V \sum_i n_i n_{i+1} \tag{1.14}$$

where $n_i = a_i^\dagger a_i$ is the occupation number at site $i$. This model cannot be solved exactly due to the interacting potential. By means of the DMRG algorithm [Sch05] with the iTensor package [FWS20], we have reproduced the phase diagram, after setting $J = \Delta = 1$, which is shown in panel (a) of Fig. 1.4, for $\mu > 0$ only since the model is symmetric for $\mu \leftrightarrow -\mu$.

The model in Eq. (1.14) is characterized by only one topological superconducting phase (TOP, yellow in the image) and four trivial phases: Topological trivial (that we denoted with TRI and the color green in the phase diagram), a Charge Density Wave sector separated into Incommensurate (I-CDW, light-blue) and Commensurate (CDW, purple) and a Schrödinger-cat-like phase (CAT, light green dashed line) which shows up at the symmetric point $\mu = 0$ as a superposition of two trivial superconducting states with different occupation numbers [HS12, TRS13, KST15, Mia17]. At $V = 0$ we recover the non-interacting case with a critical point at $\mu = 2$.

The different phases in Fig. 1.4(a) are detected from the number of edge states that appear in the chain: in the TRI, (I-)CDW and CAT phases the number of edge states is zero, while it is one in the TOP phase. The transition between I-CDW and CDW is found by looking at the homogeneity of local densities and entropy signatures [TRS13]. This phase transition is identified also when mapping the Kitaev

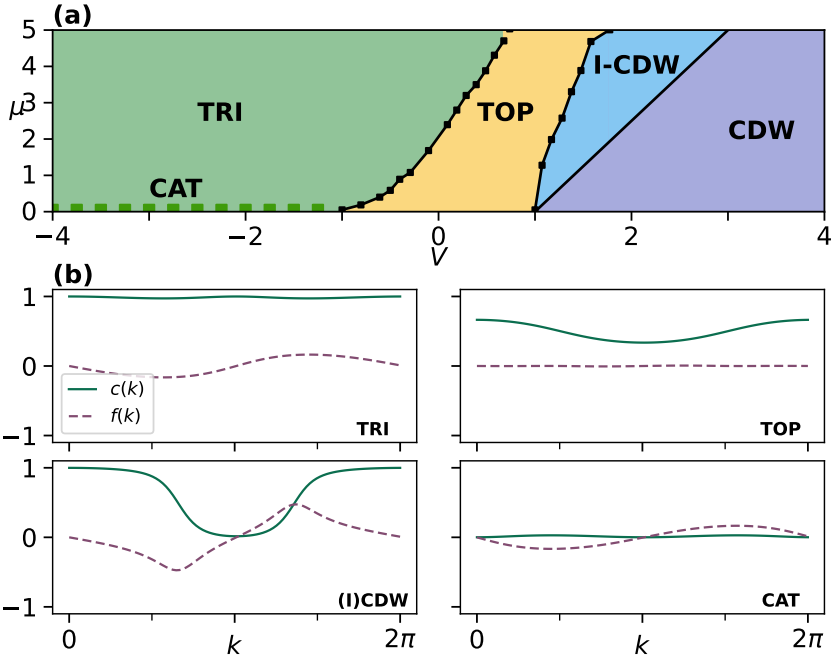



**Figure 1.4: Phase diagram of the interacting Kitaev chain.** (a) The model is characterized by one topological phase (TOP) and 4 trivial phases: the Schrödinger-cat-like phase (CAT), trivial (TRI) and (I)CDW the (Incommensurate) Charge Density Wave. (b) The correlators for each phase. We only show one correlator for the whole charge density wave sector.

chain to a Josephson junction array [HS12], in this case the transition between the phases induces a non-zero conductance due to finite-size effects. This suggests that although it is not relevant as a topological effect, this transition might be individuated in the machine learning analysis of Chapter 2.

In the interacting case, it is not possible to evaluate exactly the correlation functions $c(k)$ (Eq. (1.8)) and $f(k)$ (Eq. (1.9)) on the ground state of the Hamiltonian of Eq. (1.14). Therefore we calculate them by means of the DMRG algorithm for a lattice of size $L = 100$. Some examples of the correlation functions for the different regions of the phase diagram are shown in panel (b) of Fig. 1.4.

## 1.3 Next-to-nearest neighbour Kitaev

In Section 1.1 we studied the Kitaev model in its original form, that is with nearest neighbor coupling. It is also interesting to show the effects of a longer range coupling, specifically the next-to-nearest neighbor (NNN) coupling. Considering this type of interaction was firstly proposed by Y. Niu et al [NCH+12] as a necessary step when dealing with the realistic implementation of a quantum wire. As mentioned above, the Kitaev Hamiltonian belongs to the BDI symmetry class which is characterized by a $\mathbb{Z}_2$ homotopy group. That means that different topological phases are characterized by a different integer number which also corresponds to the number of edge states. The Kitaev chain shows the presence of one edge state which is

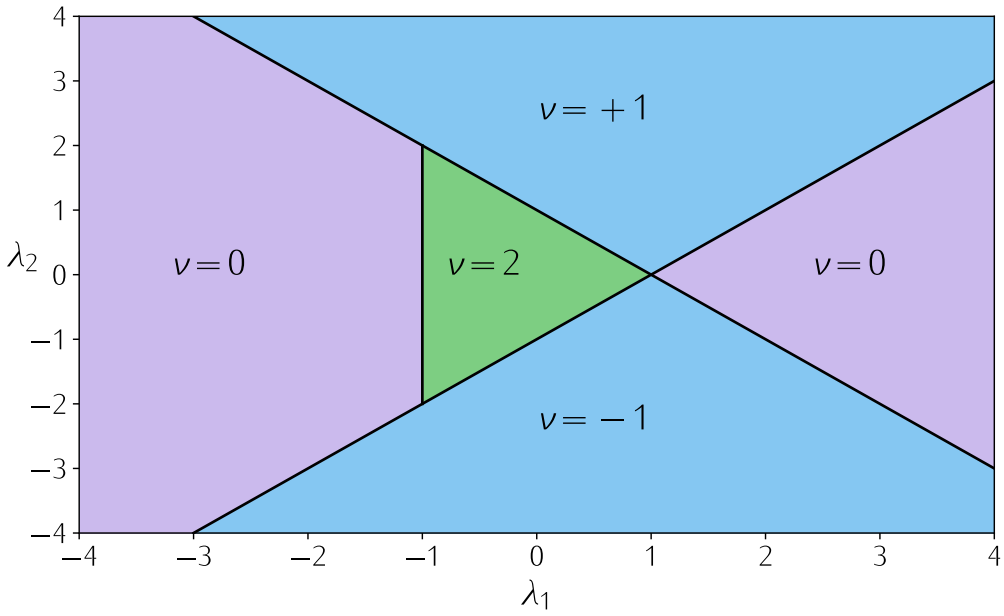



**Figure 1.5: Phase diagram of the NNN Kitaev chain**. The model of Hamiltonian (1.18) presents 5 phase characterized by a different phases identified by 3 transition lines: $\lambda_2 = \mu + \lambda_1$, $\lambda_2 = \mu - \lambda_1$ and for $\lambda_2 = -\mu$ for $|\lambda_1| < 2|\mu|$, with $\mu = 1$ in this section. Each phase a different winding number $\nu$. Blue: phases with $|\nu| = 1$. Purple: phases with $\nu = 0$. Green $\nu = 2$.

related to the winding number $\nu \pm 1$. This extended version can host up to two.

We start by writing the Hamiltonian of this extended model. We define $t_\alpha$ and $\Delta_\alpha$ as the hopping parameter and the superconductive gap of nearest neighbour ($\alpha = 1$) and next-to-nearest neighbour interaction ($\alpha = 2$). Let us set $t_1 = \Delta_1 = \lambda_1$ and $t_2 = \Delta_2 = \lambda_2$, with $\lambda_1$ and $\lambda_2$ real, so that we come to the following Hamiltonian

$$H_1 = \frac{\lambda_1}{2} \sum_{j=1}^{N-1} (c_j^\dagger c_{j+1} + c_j^\dagger c_{j+1}^\dagger + h.c.) \tag{1.15}$$

$$H_2 = \frac{\lambda_2}{2} \sum_{j=1}^{N-1} (c_j^\dagger c_{j+2} + c_j^\dagger c_{j+2}^\dagger + h.c.) \tag{1.16}$$

$$H_\mu = -\mu \sum_{j=1}^{N} (c_j^\dagger c_j - \frac{1}{2}) \tag{1.17}$$

$$H = H_\mu + H_1 + H_2 \tag{1.18}$$

Since all the parameters are real, the total Hamiltonian (1.18) still enjoys the same symmetries of the Kitaev chain with nearest neighbor coupling and thus belongs to the BDI topological class.

**Phase diagram** We can diagonalize Hamiltonian (1.18) by going into momentum space (see Section 1.1), where we obtain the single-particle functions in the form:

$$h_y(k) = \lambda_2 \sin 2k + \lambda_1 \sin k, \quad h_z(k) = \mu - \lambda_2 \cos 2k - \lambda_1 \cos k \tag{1.19}$$

and then performing a Bogoliubov transformation that gives us the energy spectrum:

$$\varepsilon(k) = \pm\sqrt{\mu^2 + \lambda_1^2 + \lambda_2^2 + 2\lambda_1(\mu - \lambda_2)\cos(k) - 2\lambda_2\cos(2k)} \tag{1.20}$$

The energy gap closes for $\lambda_2 = \mu + \lambda_1$, $\lambda_2 = \mu - \lambda_1$ and for $\lambda_2 = -\mu$ for $|\lambda_1| < 2|\mu|$. This phase transitions and the whole phase diagram are shown in 1.5 with $\mu = 1$.

To better understand the diagram 1.5, we first consider some limiting cases.

- $|\lambda_1|, |\lambda_2| \ll |\mu|$: this is the trivial phase which hosts no Majorana edge modes;
- $\lambda_2 = 0$: we regain the Nearest Neighbor coupling Kitaev chain that we studied in the last section. Thus, we can see the phase transition from 0 edge modes to 1 happening when $\lambda_1$ is larger then $\mu/2$;
- $\lambda_1 = 0$: in this case there is only next-to-nearest neighbor coupling so the wire splits into two Kitaev sub–chains (the odd and even sites). Each sub–chain has one Majorana edge mode appearing when $|\lambda_2| > \mu$.

Finally, we are interested in the correlation functions $c(k), f(k)$ (defined in the previous section (1.8),(1.9)) that will create our dataset elements:

$$c(k) = \frac{1}{2} + \frac{\mu - \lambda_2\cos 2k - \lambda_1\cos k}{2|\varepsilon(k)|} \tag{1.21}$$

$$f(k) = \frac{\lambda_2\sin 2k + \lambda_1\sin k}{2|\varepsilon(k)|}. \tag{1.22}$$

.

We considered a chain of length $L = 100$ and generated a dataset of 6000 equally spaced points in the range $\lambda_1 \in [-4, 4], \lambda_2 \in [-4, 4]$ to cover the full phase diagram of Fig. 1.5. Each datapoint is a vector made of $2L$ entries corresponding to $c(k)$ followed by $f(k)$.

## 1.4 Extended Hubbard

The fermionic Hubbard model [ABKR22, Tas97] has been deeply studied in the past due to its solvability in one dimension, which can give physical insights regarding strongly correlated electronic systems, and due to the growing possibilities of simulating it through quantum technologies [DHTPac21, HFJea17, TSP18]. In recent years extensions of this model have been interest of study in order to investigate high temperature superconductivity [RB99, DRTG22]. In the following we will consider the one-dimensional Hamiltonian [Nak00]

$$H = -t\sum_{i,\sigma=\uparrow,\downarrow}\left(a^\dagger_{i+1,\sigma}\,a_{i,\sigma} + \text{h.c.}\right) + U\sum_i n_{i\uparrow}n_{i\downarrow} + V\sum_i\sum_{l=1}^{r_c} n_i n_{i+l} \tag{1.23}$$

where $a^\dagger_{i,\sigma}$, $a_{i,\sigma}$ are the usual fermionic creation and annihilation operators for particles of spin $\sigma = \uparrow, \downarrow$ , $n_{i\sigma} = a^\dagger_{i,\sigma}a_{i,\sigma}$ and $n_i = n_{i\uparrow} + n_{i\downarrow}$. Hamiltonian (1.23) represent the famous Hubbard model with the on-site potential $U$ and an additional soft-shoulder term $V$ which covers a longer range of $r_c$ sites. We will consider a chain

of $L = 30$ sites with periodic boundary conditions (PBC) at filling $\rho = 2/5$, with $\rho_\uparrow = \rho_\downarrow = 1/5$ and interaction range $r_c = 2$. The choice of $L = 30$ is connected to the frustration of the model for these particular values of $r_c$ and $\rho$ and it will be clarified in what follows. A full description of the phase diagram of this model is way beyond the purposes of this introductory section, for this reason we limit ourselves to list the main topological phases that can be predicted with theoretical and numerical approaches showing the many limits of these methods, paving the way to the use of machine learning techniques.

**Classical Limit** This is obtained by setting $t = 0$. In this framework the determination of the $T = 0$ $K$ phase diagram is rather simple. In fact, as hopping is completely turned off, one can prove that, for fillings greater than $1/(r_c + 1)$, the classical ground state is constituted by three different blocks, which we will refer to as $A$, $B'$ and $B$ and are represented in Fig. 1.6(a). The length of $A$ and $B'$ blocks is of $r_c + 1$ sites, while block $B$'s is of $r_c + 2$ sites. The energies of these blocks are $E_A = 0$, $E_B = V$ and $E_{B'} = U$ respectively. We consider the two following limits:

- $U \gg V$: the elementary unit of this phase must have a length which contains an integer number of particles and no $C$ block. This is obtained with one block $B$ and two blocks $A$, as represented in Fig. 1.6(b). We will refer to this phase as *Nearest neighbours* phase (NN);
- $V \gg U$: with similar reasoning, the elementary unit is constituted by four blocks $A$ and one block $B'$, as represented in Fig. 1.6(b). We will refer to this phase as *Doublons* phase ($D$).

It is straightforward that the minimal length which allows for both of these phases is $L = 30$, this justifies our choice of chain length. As going from one phase to the other is equivalent to trading two $B'$ blocks for three $B$ blocks, the classical phase transition line is $U = 3V/2$. It is important to stress that the total number of blocks constituting the system changes from one phase to the other. In particular, for generic value of $r_c$, $L$ and $\rho = 2/5$ the $NN$ phase presents

$$M_{NN} = \frac{3}{5}\frac{L}{r_c} \tag{1.24}$$

blocks, while the $D$ phase has a number of blocks equal to

$$M_D = \frac{L}{r_c + 1}. \tag{1.25}$$

**Small Coupling** In the regime with low $V < U$ we can use bosonization [Gia04] to explore the phase diagram, the details of this operation can be found also in [CTEP24]. The main result is that when the soft-shoulder potential is turned off, i.e. $V = 0$, the interaction is always marginal, and the system shall therefore always be a Tomonaga Luttinger Liquid ($TLL$), which is a widely-known and studied model describing one dimensional conductors [Tom50, Lut04]. When, on the other hand, the potential is turned on, the system reaches a phase transition point for $V \simeq U$ [CTEP24, Mal13] as depicted in Fig. 1.6(c).

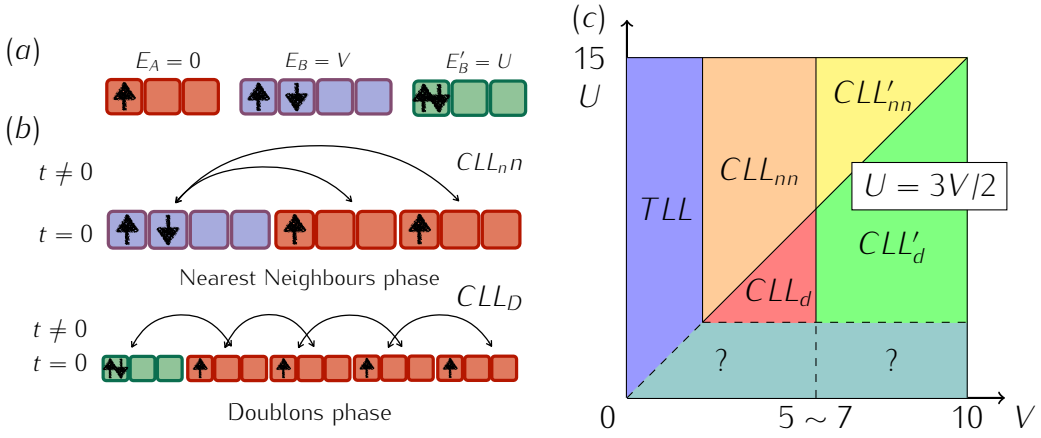



**Figure 1.6: Possible Phase Diagram of Extended Hubbard**. (a) Types of blocks of sites forming in this model. From left to right: the A block with zero energy since there is only one occupation, the $B$ block with energy contribution $V$ due to the presence of two neighbouring occupations and $B'$ with energy $U$ due to the on-site double occupancy. (b) Combination of blocks forming macro-structures that compose different ground states. Top: A combination of 1 $B$ block + 2 $A$ blocks is the main brick that composes the $CLL_{nn}$ phase. Bottom: 1 $B'$ block + 3 $A$ blocks making up the main component of the doublons phase $CLL_D$. For both structures we have the $t = 0$ in which the blocks do not move and the $t \neq 0$ scenario in which tunneling effects allow the blocks to exchange position between each other. (c) Tentative reconstruction of the phase diagram using initial assumption and the previous knowledge of the model. A more complete proposal will be given at the end of the Chapter 2.

**Intermediate $U, V$** For intermediate values of $V$ and $U$, it is possible to apply Haldane's formalism, which is also known as *phenomenological bosonization*. As it is shown in [MDLP13] applying Haldane's formalism, one ends up with an Hamiltonian with an identical functional form as $TLL$. For this reason this phase shall preserve many properties of the $TLL$ phase, with the fundamental difference that the elementary units of the system are no longer single particles, but rather clusters of particles, as represented in Fig. 1.6(b). We hypothesize the existence of two distinct $CLL$ phases, namely the Cluster Luttinger Liquid of nearest neighbours ($CLL_{nn}$), whose fundamental entities are blocks $A$ and $B$, and Cluster Luttinger Liquid of Doublons ($CLL_d$), whose fundamental entities are blocks $A$ and $B'$. For the same considerations valid for the classical limit, it is reasonable to expect the transition between the two phases to happen in the proximity of the classical phase transition line $U = 3V/2$. We expect this phases to be the ground state of the system for intermediate values of the couplings $U$, $V$, as represented in Fig. 1.6(c).

**$U \gg V \gg t$ Limit** For this part we can erase the spin degrees of freedom since double occupancies are necessarily prohibited. The results we expect on this phase can be taken by [MDLP13] which considered a similar problem and suggested that strong coupling perturbation theory led to the theoretical prediction that the system shall undergo a general *crystallization* (i.e. more localized particles) for high values

of $V$. Their prediction are accurate as far as $V/t > 10$ and break down for $V \simeq 5-7$. Assuming the transition points is $V \in [5, 7]$ we split the nearest neighbours and doublons phases between their liquid phase (denoted $CLL_{nn}$, $CLL_d$ respectively) and crystal phase: $CLL'_{nn}$, $CLL'_d$.

All in all, we put all the information gained into the phase diagram in Fig. 1.6(c) but we clearly see that there is a lot of space to investigate, much more that the other models considered in this work. We will thus heavily rely on results obtained with the machine learning methods in of Chapter 2 to obtain a clearer picture of the phases.

**Data** As we can see from the amount of phases that are present in this model and our lack of knowledge on the areas with strong interactions, we expect this model to be more difficult to tackle than the previously listed ones. In addition to that, the limited size of the model $L = 30$ – that was decided merely for costly computation reasons – would generate low-dimensional points. For these reasons we decided to generate more types of data. Specifically, we focus on charge and spin structure factors, which are defined as

$$S_c(k) = \frac{1}{L}\sum_{l,j} e^{ik(l-j)} \left(\langle n_l n_j\rangle - \langle n_l\rangle\langle n_j\rangle\right) \tag{1.26}$$

$$S_s(k) = \frac{1}{L}\sum_{l,j} e^{ik(l-j)} \left(\langle S^z_l S^z_j\rangle - \langle S^z_l\rangle\langle S^z_j\rangle\right) \tag{1.27}$$

with $k = 2\pi n/L$, $n = 0, \ldots, L/2-1$ (it is easy to see that for $n = L/2, \ldots, L$ there is only a difference in sign) and with:

$$n_i = n_{i\uparrow} + n_{i\downarrow} \tag{1.28}$$

$$S^z_l = n_{l\uparrow} - n_{l\downarrow}. \tag{1.29}$$

On the other hand we shall consider a local density operator, defined as

$$\vec{n} = (\langle n_1\rangle, \langle n_2\rangle, \ldots, \langle n_L\rangle). \tag{1.30}$$

All these quantities are numerically computed using DMRG [Sch11] onto a $L = 30$ chain with Periodic Boundary Conditions (PBC). In particular, we made use of the iTensor library [FWS20]. The convergence found for maximum bond dimension equal to 400 and we compute the ground state for $N = 4000$ values of $U$ and $V$ with 100 different values of $U \in [0, 25]$ equally spaced and 40 equally spaced values of $V \in [0, 10]$. Even if computations were worked out for $U \in [0, 25]$, plots in the following are limited to $U = 15$, as the upper part of the phase diagram does not show any relevant difference with the plotted region.

# 2 | Phase diagrams reconstruction techniques

In this chapter we show many concrete examples of how Machine Learning techniques can be exploited in a common and important task of many body physics: the reconstruction of the phase diagram of a model. Investigating and classifying the phases of a quantum many-body systems is a non-trivial job and in most cases it cannot be done analitically. For this reason many approximation methods have been developed in the years. The most successful one is certainly the DMRG algorithm which finds the groundstate of a system from which correlation functions and order parameters can be calculated approximately but still with a very good precision.

Yet, we often find interacting systems that are difficult to simulate even with the DMRG and can be resource-demanding. For this reason, in this chapter we show how unsupervised and supervised learning algorithms can be trained to recognize the phases of a solvable non-interacting model and then used to perform the classification on the same system but in presence of interactions.

For each algorithm we first give an intuitive explanation of its functioning and we mostly focus on how its results can be interpreted in order to give a re-usable instrument to investigate the topological phases of a model. We then proceed to apply it to the systems mentioned in the previous chapter with two types of training. The standard one means training and testing on the data of the same model (interacting or not). The second one, when possible, consists in training first to a non-interacting version of the model and then to the interacting version. We refer to the former as transfer learning as it is typically done in machine learning [Ben12]. The content of this chapter is redacted from and extends the work we published on this topic during my PhD studies [TMVE23] and another one that is currently in preparation [CTEP24].

## 2.1 Principal Component Analysis

Principal Component Analysis (PCA) is a simple algorithms used in statistics and Machine Learning to reduce the dimension of a dataset. This is done by extracting the (orthogonal) directions along which the dataset varies the most and projecting the data onto these vectors. Figure 2.1 shows a 3D qualitative example of a dataset made with correlators like the ones we obtained in Chapter 1. In this section we want to explore how much information PCA can extract from the correlator data and

what it can tell us about the topological phases of the system. We will see soon that it is a non–trivial amount of information.

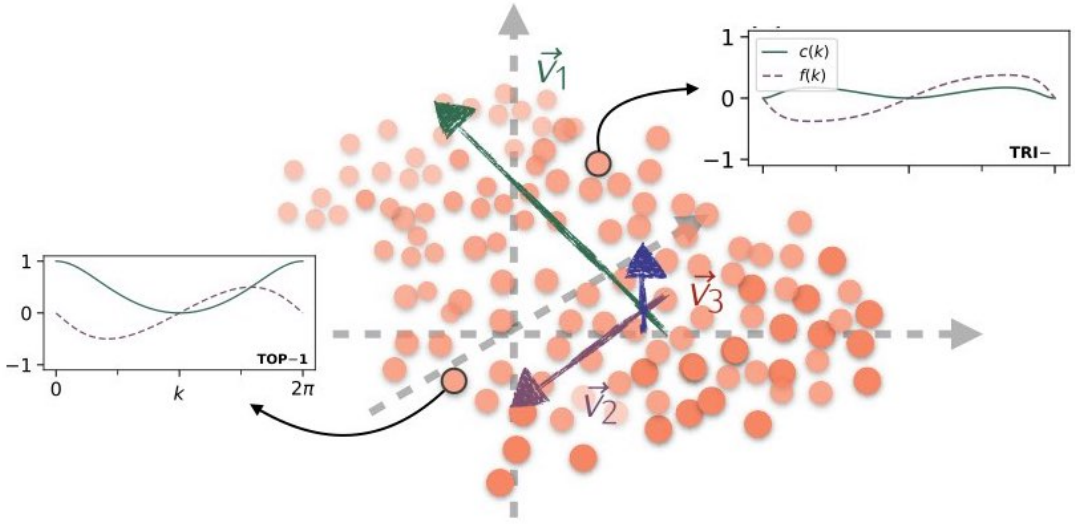



**Figure 2.1: Principal Component Analysis**. The data in this simplified scenario is mostly distributed on a plane. The vectors $\vec{V}_1, \vec{V}_2, \vec{V}_3$ are the principal components, that is the vectors obtained from PCA, and we can see they are aligned with the directions of the plane because those are the directions more informative about our data. Each vectors is also represented by an eigenvalue that quantifies how much information is encoded in each of the directions. In this case $\vec{V}_1, \vec{V}_2$ will have larger eigenvalues $\lambda_1, \lambda_2$ (in modulus) compared to the third direction $\vec{V}_3$ which is not representative of the data.

In order to apply PCA we start by creating a *design matrix*

$$X = \begin{pmatrix} c_1(k_0) & \dots & c_1(k_{L-1}) & f_1(k_0) & \dots & f_1(k_{L-1}) \\ & & \vdots & & & \\ c_N(k_0) & \dots & c_N(k_{L-1}) & f_N(k_0) & \dots & f_N(k_{L-1}) \end{pmatrix}. \tag{2.1}$$

Each row is formed by the correlation functions $c(k)$ and $f(k)$ of the model calculated for one of the $N$ points of the phase diagram. For each model we generated $N$ pairs of points $(x_\alpha, y_\alpha)$, $\alpha = 1, \dots, N$ where $x, y$ represent two Hamiltonian parameters. Each column represents one of the Fourier components of the correlation functions with quasi-momentum $k_n = 2\pi n/L$ $(n = 0, \dots, L-1)$ that are interpreted as the features of the data from which the PCA extracts the principal components. We rescale the columns of $X$ such that they have zero mean and unit standard deviation and we compute the eigenvalues $\{\lambda_i\}$, the explained variances $\epsilon_i = \lambda_i / \sum_j \lambda_j$, and the eigenvectors $\{\mathbf{w}_i\}$ of the correlation matrix $\mathcal{S} = X^T X$. The principal components of the data $X$, which are the eigenvectors of $\mathcal{S}$ corresponding to the largest eigenvalues, are the directions in a $2L$ dimensional space along which the original data shows the largest variance. The first $d$ of these vectors are generally used to project the data into a smaller number of features. A common measure of the projection along the principal components used in the context of detection of phases is the *quantified*

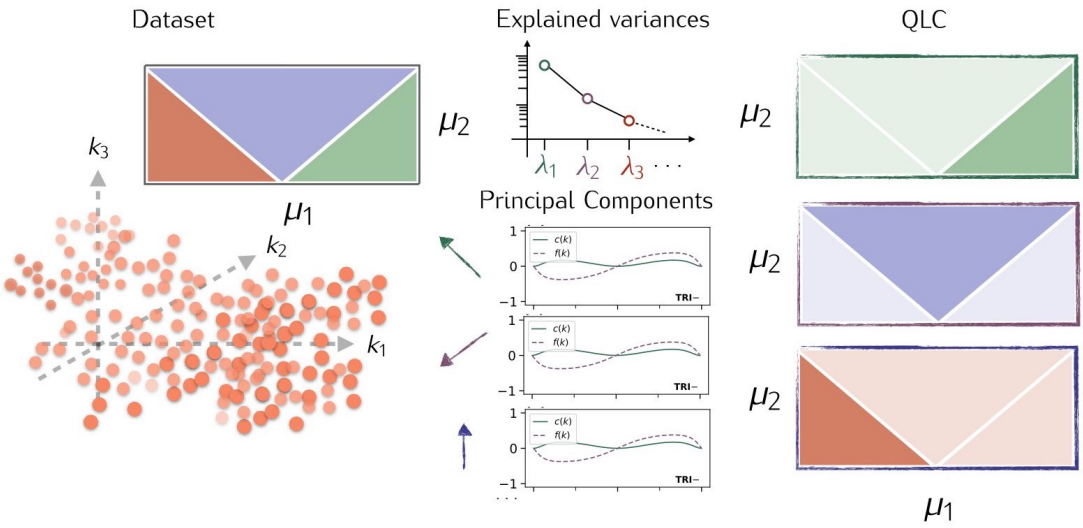



**Figure 2.2: PCA for topological phase detection**. Schematic representation of our method. We start with a dataset of points in $2L$ dimensions obtained from some phase diagram that depends on two parameters $\mu_1, \mu_2$ (left). We extract the first (largest) explained variances and corresponding principal components (center), they are graphically represented as vectors (cf. Fig 2.1). Finally, we project the data on these components (calculate the QLCs) which might highlight different sets of the phase diagram (right).

*leading component* (QLC) [Wan16, HSS17]. For each model, we compute the QLC by dividing the set of the two parameters in 40 sections, creating a grid of $40 \times 40$ subsets, each made of $M$ datapoints, $M$ depending on the number of points $N$. Then, for each of the subsets we calculate the quantity

$$p_i = \sum_s \frac{|X_s \cdot \mathbf{w}_i|}{M} \tag{2.2}$$

where $s$ runs on the elements of each subset, $X_s$ is the $s$-th row of the matrix $X$ corresponding to the $s$-th point of the subset, and $\mathbf{w}_i$ is the $i$-th eigenvector of $\mathcal{S}$. The value of the QLC for a datapoint shows how much that point is represented by that specific principal component.

Figure 2.2 shows an example of this simple pipeline and how it allows us to extract different phases from a topological model. We start from a dataset of correlators obtained by varying two parameters of a Hamiltonian, the example of the image shows a phase diagram that is split in three phases. We extract the eigenvalues (explained variances) and eigenvectors (principal components) of the correlation matrix. Finally we project the original data on the principal components and plot the results.

### 2.1.1 Kitaev Chain

For the Hamiltonian of Eq. (1.1), we create the design matrix $X^{(\mathrm{NI})}$ from data points generated in the range $\mu \in [-8, 8[$ and $\Delta \in [-2, 2[$ with a step of 0.1 for $\mu$ and

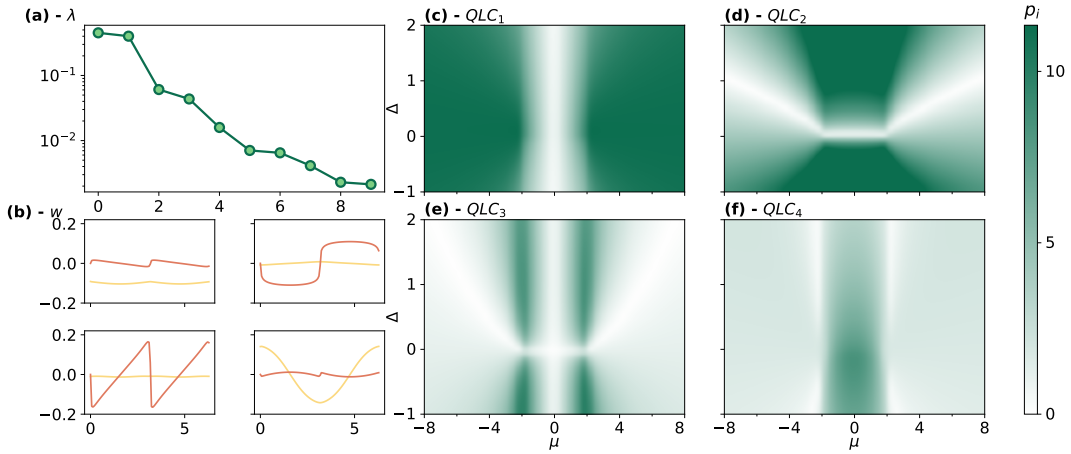



**Figure 2.3: PCA of the non-interacting Hamiltonian.** *(a) Explained variances of the first ten principal components. (b) The first four principal components* $\mathbf{w}_i$ *(*$i = 1, 2, 3, 4$*) extracted from data plotted in the same fashion of the correlators in the dataset. In red the first 100 elements of each vector and in orange the second (orange triangles) of the corresponding principal components . On the right side we have the quantified leading components (QLC)* $p_i$ *which measure the projections of the data onto the space of each of the first four principal components in (b), corresponding to the eigenvalues with largest explained variance* $\epsilon_i$*. (c) The first QLC with* $\epsilon_1 = 0.451$ *reveals the points of the phase diagram with trivial winding number. (d) The second QLC, with* $\epsilon_2 = 0.421$*, highlights the points of the phase diagram with non-trivial winding number, (e) The third QLC, with* $\epsilon_3 = 0.052$ *shows the phase transition lines, (f) The fourth QLC with* $\epsilon_4 = 0.042$ *has a high projection on the phase with* $\nu = -1$*.*

0.05 for $\Delta$, for a system with $L = 100$ sites. This subdivision of the ranges of $\Delta$ and $\mu$ corresponds to have a total of $N = 12800$ data points. The sum $\sum_{i=1}^{4} \epsilon_i$ of the explained variances (plotted in Fig. 2.3(a)) of the first four eigenvalues results in 96.6% meaning that most of the information of the data $X$ is contained in the first four eigenvectors (Fig. 2.3(b)). For this reason, in Fig. 2.3(c-f), we show the first four QLCs. These are calculated as in Eq. (2.2) by grouping the $N = 12800$ datapoints in $40 \times 40$ subsets corresponding to $M = 8$ datapoints per subset.

The explained variances $\epsilon_i$ of each eigenvector are reported for each of the first four components. In panel (c) we see that $p_1$ is large for the points with $|\mu| > 2$, this means that the first principal component allows us to find the points of the phase diagram with winding number $\nu = 0$ (that belong to the trivial phases TRI− and TRI+). This is due to the shape of the first principal component $\mathbf{w}_1$ (depicted in Fig. 2.3(b)) that resembles the shape of the correlation functions $c(k)$ and $f(k)$ shown in Fig. 1.2(b) (TOP+1 phase). In Fig. 2.3(d), we see that, instead, $p_2$ allows us to extract the points of the phase diagram with winding number $\nu = \pm 1$ as the shape of the second principal component $\mathbf{w}_2$ (Fig. 2.3(b)) is similar to the correlation functions $c(k)$ and $f(k)$ shown in Fig. 1.2(b) (TOP$\pm$1 phases).

This analysis shows that the first two principal components are sensitive to the

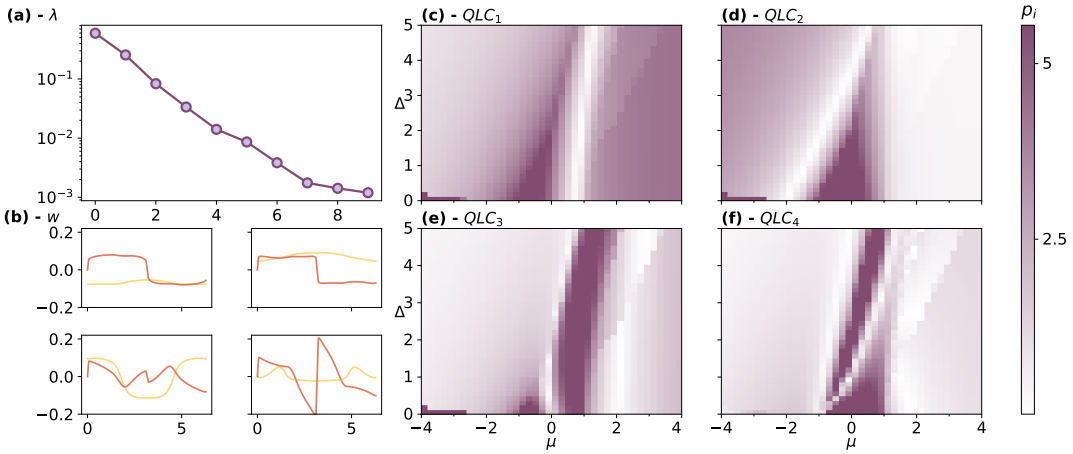



**Figure 2.4: PCA of the interacting Hamiltonian.** *(a) Explained variances of the first ten principal components. (b) The first four principal components* $\mathbf{w}_i$ *(*$i = 1, 2, 3, 4$*) extracted from data plotted in the same fashion of the correlators in the dataset. On the right side we have the quantified leading components (QLC)* $p_i$ *which measure the projections of the data onto the space of each of the first four principal components in (b), corresponding to the eigenvalues with largest explained variance* $\epsilon_i$*. (c) The first QLC with* $\epsilon_1 = 0.595$ *reveals the points of the phase diagram with trivial winding number. (d) The second QLC, with* $\epsilon_2 = 0.255$*, highlights the points of the phase diagram with non-trivial winding number, (e) The third QLC, with* $\epsilon_3 = 0.083$ *shows the phase transition lines, (f) The fourth QLC with* $\epsilon_4 = 0.033$ *has a high projection on the phase with* $\nu = -1$ *and is the most effective at recognising the transitions between incommensurate and commensurate CDW.*

trivial and non-trivial phases. All the other 198 QLCs have a total explained variance of the order of 13%, in particular the third QLC (Fig. 2.3(e)) has explained variance $\epsilon_3 = 5.2\%$ and recognizes the phase transition lines, while the fourth (Fig. 2.3(f)) has explained variance $\epsilon_4 = 4.2\%$ and has a larger projection on the phase with winding $\nu = -1$.

### 2.1.2 Interacting Kitaev chain

We repeat the analysis with the data obtained from the interacting Hamiltonian. To this end, we construct a design matrix $X^{(I)}$ with data obtained from the interacting Hamiltonian in the range $\mu \in [0, 5[$, $V \in [-4, 4[$ with a step of 0.1. We plot in Figure 2.4(a) the first ten explained variances and we see that they decrease exponentially. The first four in this case amount to 96.7% of the total, the values of each one are described in the caption of Figure 2.4. The main result of this analysis is being able to capture the phases of the model. In particular the central topological phase around $\mu \sim 0$ seems to be highlighted by each QLC in different ways. In particular we can clearly separate it from the trivial and the charge density wave (CDW) phases (TRI and CDW sectors of Fig. 1.2(b)). The first three QLCs – Fig. 2.4(c-e) – can also highlight the CAT phase at $\Delta = 0$ with ease while QLC number 1,3 and 4 evidently signal the phase transition between Incommensurate and Commensurate CDW.

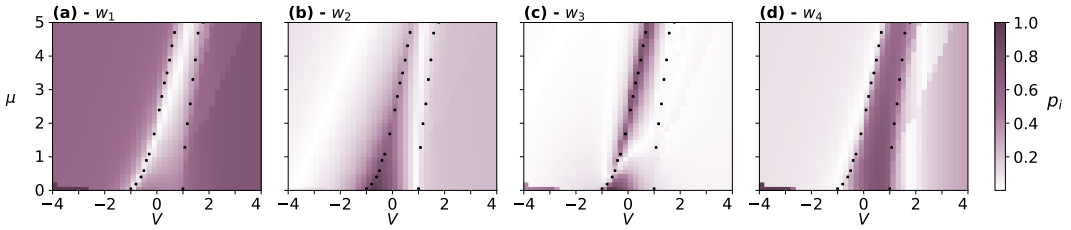



**Figure 2.5: PCA transfer learning on the interacting Hamiltonian** Projection of the interacting correlation functions along the principal components obtained applying PCA to the non interacting dataset. Darker colors correspond to areas of the phase diagram which are more similar to the principal component $\mathbf{w}_i$ ($i = 1, 2, 3, 4$) used for the projection. QLCs drawn in (a) and (d) highlight the trivial (superconduting and charge density wave) and topological phases respectively. In both cases, the CAT phase is highlighted by higher values of $p_i$ compared to the other phases. Component (c) pins down a phase transition line whereas component (b) does not seem to be informative on the interacting dataset. The projections of all four plots are rescaled independently. The dots are the same as panel (c) of Fig. 1.2, added to help recognize the phase transition points.

**Transfer learning on the interacting Hamiltonian** We are interested in understanding if the principal components of the non-interacting model computed before can be used to distinguish among the phases of the interacting Hamiltonian. In order to do that we consider again the design matrix of the interacting data. Then, we calculate the QLC by projecting these data along the principal components $\mathbf{w}_i$ of the non-interacting Hamiltonian. The first four QLC are shown in Fig. 2.5. The first (panel (a)) and the fourth (panel (d)) principal components $\mathbf{w}_1$ and $\mathbf{w}_4$ of the non-interacting data show higher projections on the interacting data and are thus able to discriminate between the trivial and topological phases, respectively. The third principal components $\mathbf{w}_3$ (panel (c)) recovers only the transition line between the trivial and the non-trivial superconducting phases while the second one seems to highlight only the region for low $\mu$ and $V \in [-1, 0]$. We note that, even though the CAT phase is present only on the single line $\mu = 0$, three out of four principal components are able to recognize it (Fig. 2.5(a), (c), (d)) and once again the phase separation inside the CDW sector is evident in the first and fourth components.

### 2.1.3 Kitaev with NNN

We repeat the same analysis to the Kitaev model with Next–to–nearest neighbour hopping. We generated a dataset of 6000 points in the range $\lambda_1 \in [-4, 4], \lambda_2 \in [-4, 4]$. The correlators span the 5 phases of the model with $\nu = 0, \pm 1, 2$, as shown in 1.3. We followed exactly the same steps as in the previous section and obtained fairly comparable results, we will thus limit our discussion in this section. In particular Figure 2.6 shows that the QLCs obtained with the PCA analysis can highlight all the phases, in order (c): $\pm 1$, (d): 0, (e): 2. Once again, the first component that is not sensitive to a specific phase shows us the phase transition lines 2.6(f).

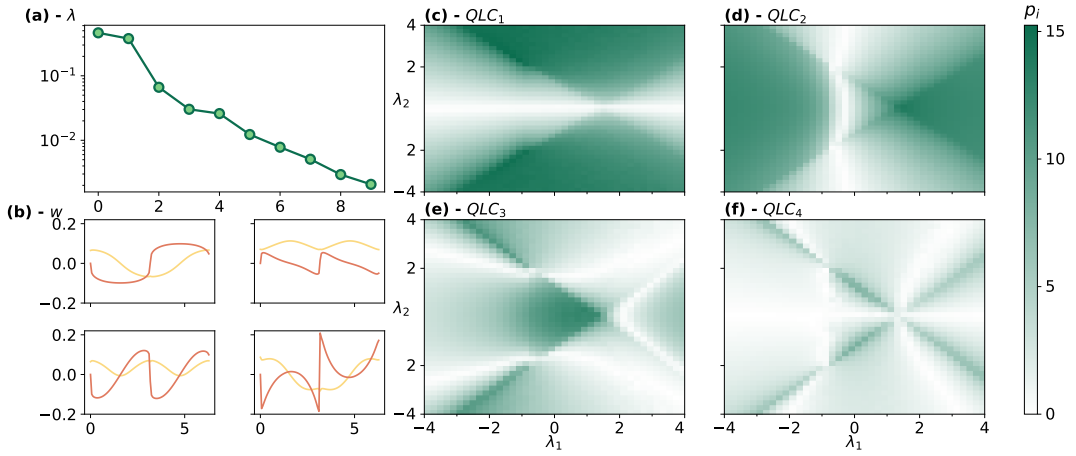



**Figure 2.6: PCA of NNN model**. (a) Normalized eigenvalues of PCA showing two relevant direction in the dataset correlations, (b) Eigenvectors of PCA. (c-f) Projections of data onto the first four eigenvectors highlighting the phases of the model.

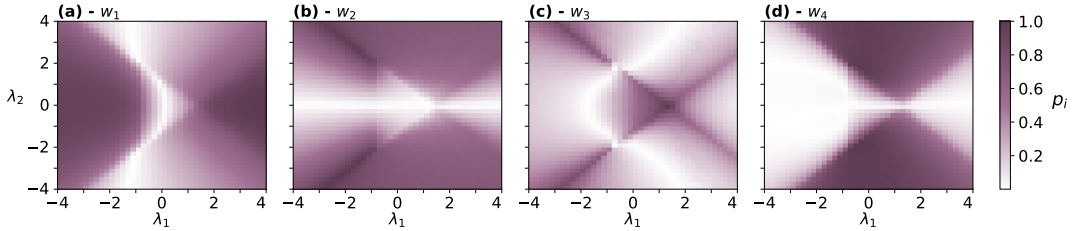



**Figure 2.7: PCA transfer learning on NNN model**. PCA projections of the NNN Kitaev data onto the Kitaev chain eigenvectors correctly predicting the phases of the model.

**Transfer learning** To prove once again the level of versatility of PCA we apply transfer learning by using the principal components extracted with PCA on the non–interacting dataset and we apply it to the NNN dataset. We consider only the first four principal components as they contain almost 100% of the explained variance of the Kitaev chain and we project the correlators of the NNN model onto them. The projections are shown in Figure 2.7 and it is evident that all the phases of the model can be directly detected.

We can say with fairness that the PCA allows us to learn the underlying pattern which distinguishes a trivial phase from a topological one in the Kitaev model. This is a good indication that even a supervised method can exploit the representation it learns of the non-interacting data to classify the interacting ones.

### 2.1.4 Extended Hubbard

In order to apply PCA, we arrange the data in a *design matrix* which differs from the other models only because we are concatenating different observables: the spin and charge structure factors (eqs. 1.26) and the local density operators (eq (1.30)) explained in Section 1.4. We note again that the reason for this variety of data is

due to the expected difficulty at recognizing the phases of this model. Considering that also the analytical and numerical theory does not currently go beyond certain threshold values (see Section 1.4). The dataset was also produced for a short chain of $L = 30$ for two reasons. Firstly, it is computationally hard to obtain data with larger system size and secondly, as explained in Section 1.4, for our choice of filling $\rho = 2/5$ the length $L$ must be a multiple of 10 and 15.

$$X^H = \begin{pmatrix} \vec{S_c}^{(1)} & \vec{S_s}^{(1)} & \vec{n}^{(1)} \\ \dots & \dots & \dots \\ \vec{S_c}^{(N)} & \vec{S_s}^{(N)} & \vec{n}^{(N)} \end{pmatrix} \tag{2.3}$$

The index indicates the point of the phase diagram where the observables are evaluated and

$$\vec{S_\alpha} = \big(S_\alpha(k_0), S_\alpha(k_1), \dots, S_\alpha(k_{L/2-1})\big) \tag{2.4}$$

with $\alpha = c, s$ and $k_n = 2\pi n/L$.

For this model we generated 4000 points of the phase diagram in the range $U \in [0, 20]$ and $V \in [0, 10]$, $k_n = 2\pi n/L$ with $n = 0, \dots, L/2 - 1$ and $L = 30$. The design matrix has shape $4000 \times 60$ (where the 60 columns are made of 15 values of $S_c(k_n)$, 15 values of $S_s(k_n)$ and 30 values of $n(x_i)$), for clarity. So the eigenvalues and vectors will also have length $2L$ as with he previous models, the rest of the procedure is in fact identical.

The relative weights of the first ten eigenvalues (in decreasing order of magnitude) of $\mathcal{L}$, which are defined as

$$\lambda_i = \frac{w_i}{\sum_i w_i} \tag{2.5}$$

are reported in Fig. 2.8(a). The index $i$ runs from the biggest eigenvalue to the smallest one. The first ten eigenvalues sum up to almost 98% of the total.

Next, we plot the QLCs $P^{(i)}$, which can be seen as the projection of the principal components onto the phase diagram under examination. The plots of $P^{(1)}$, $P^{(2)}$ $P^{(5)}$ and $P^{(7)}$ are represented in Fig. 2.8. From the projection of the first eigenvector $P^{(1)}$ we can clearly see that the dataset experiences a drastic variation for $V \simeq 5.5$. This trend is confirmed by all the projections under considerations and it coincides with the theoretical predictions from perturbation theory (see [CTEP24]). In addition to that, the region of the phase diagram delimited by $U \simeq 5$ and $U \simeq 3V/2$ seems to present radical differences w.r.t. the rest of the phase diagram.

$P^{(1)}$, $P^{(2)}$ and $P^{(5)}$ also suggest that, for $V \simeq 5.5$, there is a significant change in the observables when $0 < U \lesssim 2$ and $2 \lesssim U \lesssim 5$. Other projections with relevant weights such as $P^{(3)}$, $P^{(4)}$ and $P^{(6)}$ exhibit similar patterns and are therefore not represented. Finally, $P^{(7)}$ underlines one tiny region of the phase diagram, whose nature will be better understood with the $k$-means analysis of section 2.2. For the time being it is only worth noticing that this region sits almost at $U \simeq 3V/2$.

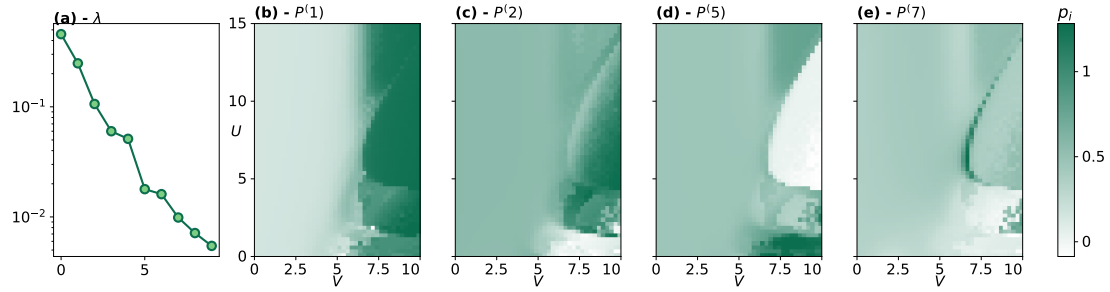



**Figure 2.8: Plots of the projections**. Projections $P^{(1)}$, $P^{(2)}$, $P^{(5)}$ and $P^{(7)}$, described in the text, are plotted here. The first three characterize the $V \gtrsim 5.5$. region, with differences arising for $0 < U \lesssim 2$, $2 \lesssim U \lesssim 5$ and $5 \lesssim U \lesssim 3V/2$. The last one, i.e. $P^{(7)}$, underlines one small region of the phase diagram in correspondence of $U \simeq 3V/2$.

## 2.2 $K$-means

$K$-means clustering is a machine learning method for finding clusters and cluster centers in a set of unlabelled data [Mac03]. The algorithm starts by choosing a number of cluster centers called *centroids* and then it iteratively moves the centers to minimize the total variance within each cluster. The centroids are the central point of every cluster that are calculated by the algorithm autonomously, so we interpret them as the most representative points of the cluster. Figure 2.9 shows a qualitative example of this process.

More specifically, the algorithm starts by randomly partitioning the $N$ elements of the dataset into $K$ subsets $P_1, \dots P_k$. In this way we can build the matrix $U \in \mathbb{N}^{K \times N}$ whose entries numbers are 0 or 1 according to which partition each point was assigned to. At the same time a set of centroids $C = C_1, \dots C_k$ is created also randomly. The loss function to minimize is simply given by:

$$\mathcal{L}(U, C) = \sum_{i=1}^{K} \sum_{X_j \in P_i} ||X_i - C_k||^2 \tag{2.6}$$

Setting $U_0, C_0$ the initial partition and centroids, $k$-means iteratively solves the minimizations:

$$U_{i+1} = \min_{U} \mathcal{L}(U, C_i) \tag{2.7}$$

$$C_{i+1} = \min_{C} \mathcal{L}(U_i, C) \tag{2.8}$$

until convergence which is usually reached either when the matrix values of $U$ do not change or the variation of $\mathcal{L}$ is below a certain threshold. The output of the algorithm is one label for each point assigning it to a cluster.

The choice of the number of partitions $K$ is clearly crucial in this algorithm. In order to settle on a value of $K$ without any *a priori* knowledge of our data we choose to calculate the *silhouette score* $\widetilde{S}$ which is a value representing the average quality of the clusterization for each $K$. Calling $a_i$ the mean distance of a sample point to

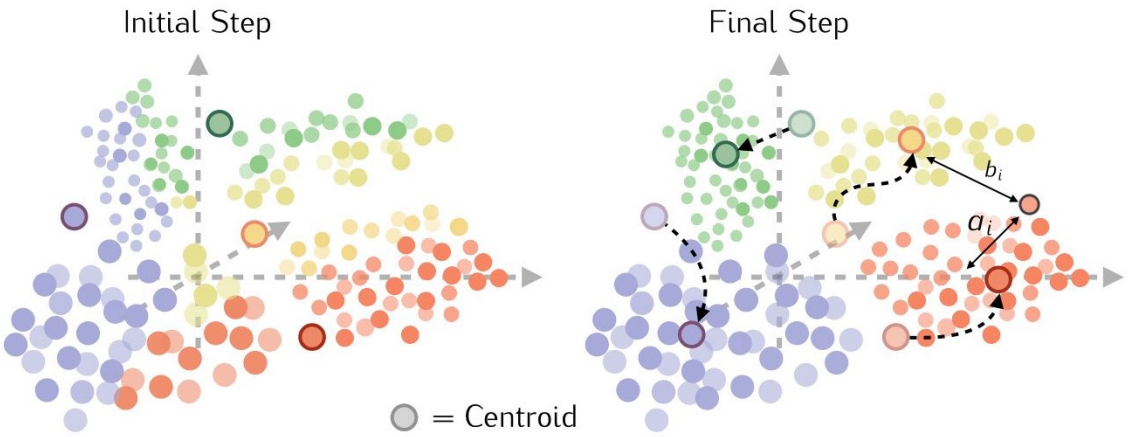



**Figure 2.9: $k$-means**. Visual representation of the clustering operation performed by $k$-means. At the initial step the points of the dataset are assigned to the closest of $K$ random cluster centers called *centroids*, in this case 4. After the final step the centroids have been moved in order to minimize the variance of each cluster. The image also shows an example of the $a_i$ and $b_i$ average distances needed to calculate the silhouette coefficient of a point.

all the points in its cluster and $b_i$ its mean distance to all the points of the second closest cluster (see Fig. 2.9) we can define and calculate the silhouette coefficient of a data point $S_i$ and the silhouette score of the dataset $\tilde{S}$ as

$$S_i = \frac{(b_i - a_i)}{\max(a_i, b_i)} \tag{2.9}$$

$$\tilde{S} = \frac{1}{N} \sum_i S_i \tag{2.10}$$

where $i$ runs over all the $N$ elements of the dataset. The values for $S$ lie in the range $[-1, 1]$ where negative numbers correspond to a wrongly assigned point. Positive values indicate the right classification and the quality increases reaching 1. Points with a silhouette close to 0 belong to overlapping clusters.

The information gained from the silhouette is thus crucial to allow us a deeper analysis of the topological phases respect to one offered only by the $k$-means labels assigned to every point. In fact, the latter simply force every point into a cluster. The former, instead, gives us an estimation of how well a point fits into its cluster, if this value is low it means that the algorithm is indecisive about two possible classes and by definition that could be a phase transition point. This intuition that we developed from our work here [TMVE23] has proven very effective in every model studied so far.

Another meta-analysis is related to the centroids. Once they are identified, we can gain more in-depth analysis of how the data points are distributed by looking at their shape. We believe this might help us find a meaning, if any, in the clusterization of the data points similarly to what we did in the PCA cases. In fact, also for the $k$-means, a centroid is a $2L$ vector that can be depicted as a set of two correlators.

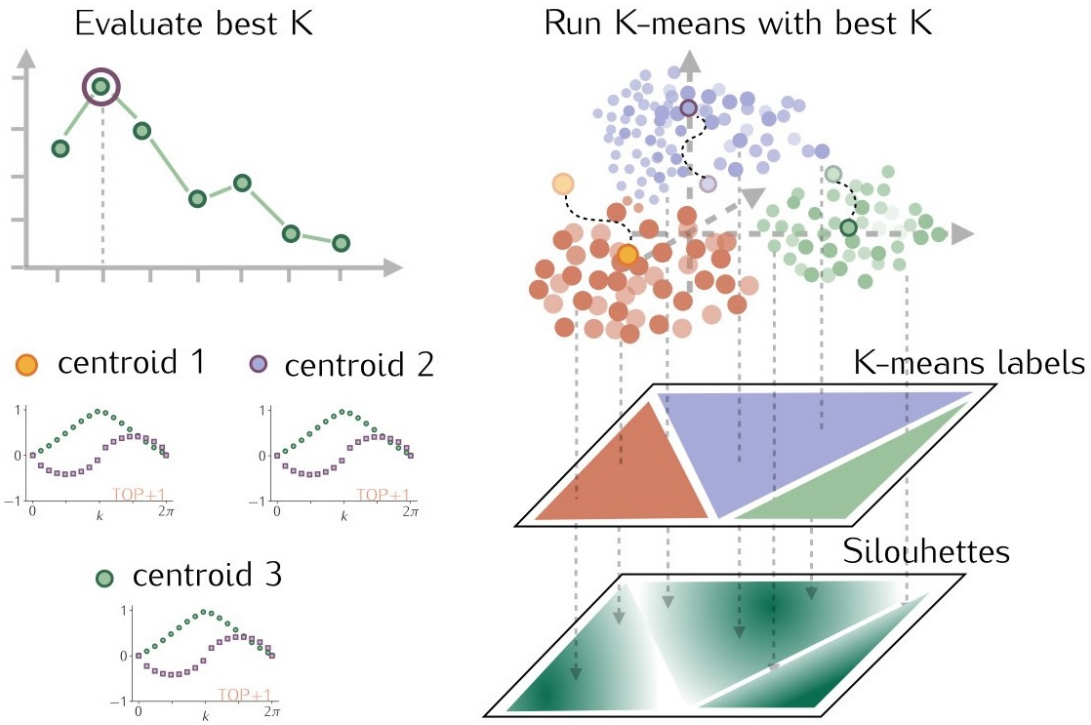



**Figure 2.10: Pipeline of phase extraction with $k$-means**. We start by evaluating the best number of cluster $K$ from the silhouette scores $\tilde{S}$. Next, $k$-means is applied from which we can obtain the $k$-means labels and the silhouette coefficient of each point that we plot back into the phase diagram. Next we analyse the shape of the centroids to see what characterize each phase found by the algorithm.

One last consideration about the results: being $k$-means an algorithm inclined to fall into local minima [AV06] it is extremely sensible to initial conditions. Thus, we run $k$-means 10 times for each $K$ and then collect the average silhouette value $\widetilde{S}$ of every point over the 10 runs, whereas the centroids and projections correspond to the largest silhouette only.

We summed up the pipeline of our application of $k$-means in order to clarify the ideas. This is explained more in-depth in the caption of Figure 2.10.

### 2.2.1 Kitaev Chain

The analysis of the silhouette for the non-interacting data of the Kitaev chain turns out to be very informative. Firstly, we find the maximum value of the silhouette score at $K = 4$ as shown in Fig. 2.11. This suggests that the most reasonable way to divide the data points is in 4 classes, which might correspond to 4 different phases. Secondly, we see that the silhouette of the points indeed can be used for identifying the phase transition lines. We confirm in fact that the points lying near a phase transition, which show properties of both phases, have silhouette value close to 0, indicating that their classification might be non ideal. We see this in Fig. 2.11(b) which shows the average silhouette of every point calculated over 10 iterations of

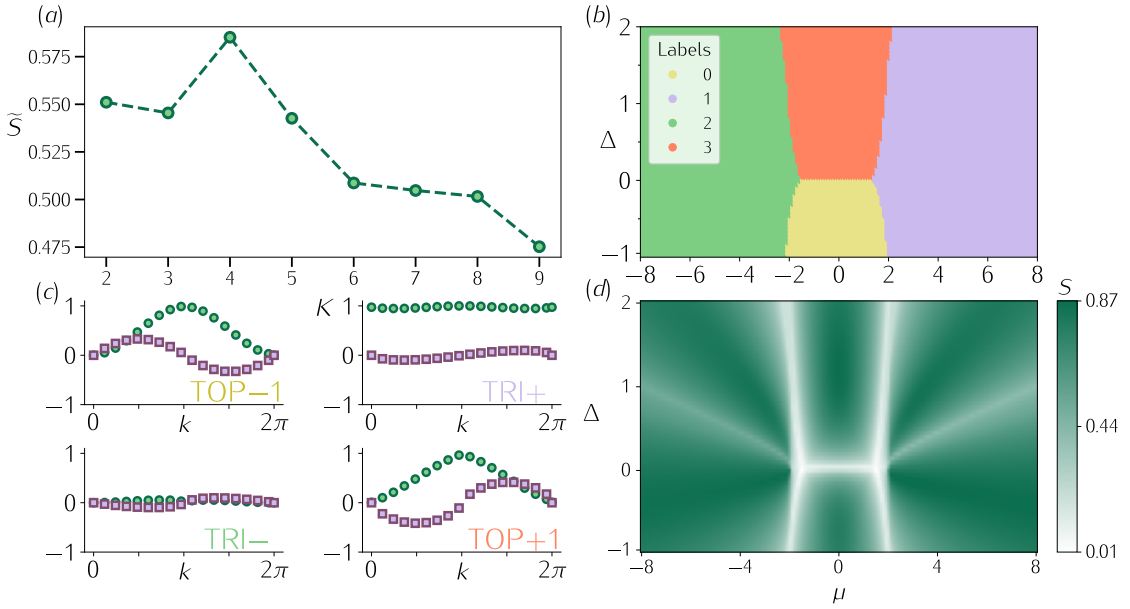



**Figure 2.11: $K$-means of non interacting data**. (a) Silhouette score for different cluster numbers $K$ suggesting 4 clusters. (b) Centroids of $K$-means. (c) Labels assigned to points by $K$–means. (d) Silhouette coefficients of each point highlighting the phase transition lines.

$k$-means algorithm applied with $K = 4$, as suggested by the previous analysis. We can see that points that are inside the phases have a larger silhouette than points lying at the phase transition.

We now consider the analysis of the centroids obtained by applying the $k$-means algorithm with $K = 4$. In Fig. 2.11 we plot: (a) the phases associated to each point by the $k$-means algorithm, a quick comparison to the original phase diagram 1.2(a) shows that the reconstruction is fairly similar; (b) the 4 centroids. Each centroid seems to exactly represent the features of the datapoints, i.e. the correlation functions, of the four different regions of the phase diagram. Moreover, they can be useful to detect better the phase transition lines.

As a final comment, the lines appearing peculiar but unfortunately we do not have a clear understanding of the behaviour of the silhouette at these points. This could probably be due to the change of sign of $\Delta$ which affects the shape of the $f(k)$ correlators. Nonetheless, the values of the silhouettes along those lines are around 0.5 so still very far from 0, which excludes a possible phase transition.

### 2.2.2 Interacting Kitaev chain

The same analysis can be repeated on the interacting dataset, obtaining similar results which are collected in Fig. 2.12. In particular the silhouette reaches its peak at $K = 4$ corresponding to the four phases of the interacting model shown in Fig. 1.2(c). This is, of course, if we associate the CDW sector as one whole phase, to which $k$-means assigned the "1" label. Yet, we see that the separation between I-CDW and CDW is present in the silhouette coefficients plotted in panel (d) of the

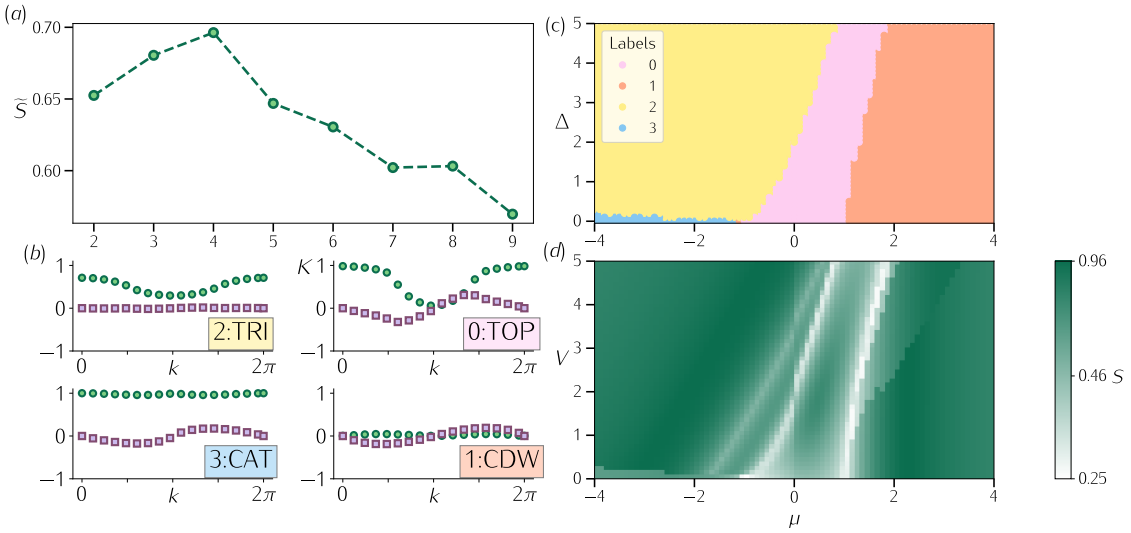



**Figure 2.12: $k$-means interacting phase diagram reconstruction.**(a) Silhouette score for different cluster numbers $K$ suggesting 4 clusters. (b) Centroids of $K$-means each one labelled by its corresponding phase. (c) Labels assigned to points by $K$–means. (d) Silhouette coefficients of each point highlighting the phase transition lines.

same figure. In this case the change of phase is not signaled by the vanishing of the silhouette but with a discontinuity in the silhouette values, much alike the one that can be seen at low $V$ and $\mu < 0$ for the CAT phase.

This results confirms that it is more informative to look at the silhouette coefficients rather than at the mere classification of the algorithm because it gives a richer interpretation of the possible phase transition lines.

**Transfer learning** In the same fashion as PCA we now apply $k$-means to the non–interacting dataset and use it to predict the phases of the interacting one. In this case we can select the number $K$ by training the clustering on the first dataset and use it to obtain the silhouette coefficients and score of the second one.

This operation is not standard in the context of $k$-means since it was meant to be trained and tested always on the same data. This is confirmed by the fact that trying our transfer learning procedure returns us the same silhouette score of 0.55 for every $K$ in the range $[2, 10]$ which suggests us that it is not an ideal value to keep track of. Therefore, instead of the silhouette score we check the average silhouette coefficient, we care to specify that the two values are related to each other but not necessarily identical. Interestingly, the average silhouette coefficient seems to suggest a possible clustering. In fact we obtain that it is maximized for $K = 2$ and reaches a second maximum at $K = 5$. The average silhouette coefficient for different $K$ are shown in Figure 2.13 along with the silhouette coefficient of every point for $K = 2$ in panel (b) and for $K = 5$ in panel (c).

For $K = 2$ we see that there is a neat separation of the topological phase from the trivial one, as well as the evidence for the CAT phase. For $K = 5$ we obtain a richer phase diagram with all the phases of the interacting model being present

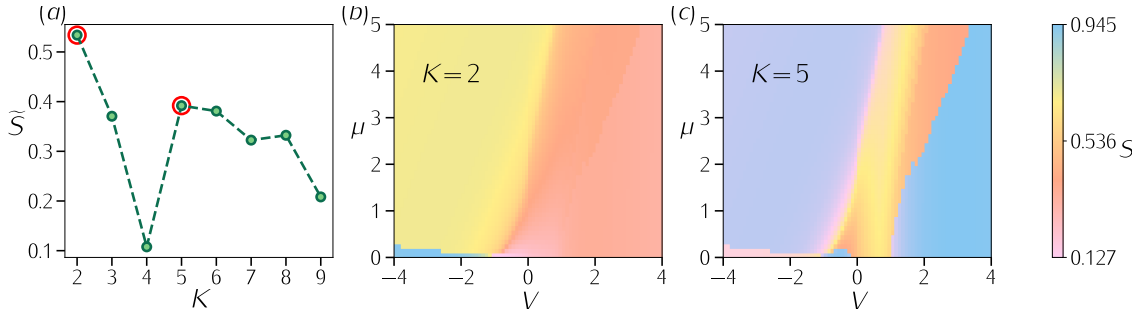



**Figure 2.13: $k$-means transfer learning.** The algorithm is trained with non-interacting data and tested on the interactin data. (a) Average Silhouette coefficients for different cluster numbers $K$ suggesting 2 and 5 clusters for classification. (b) Silhouette coefficients for $K = 2$. (c) Silhouette coefficients for $K = 5$.

and a particular sharp transition between $CDW$ and $I - CDW$. Thanks to transfer learning we are able to confirm the presence of a transition between the two charge density wave phases.

### 2.2.3 NNN Kitaev

The $k$-means analysis turns out to be very successful also on the extended Kitaev model with next-to-nearest neighbour hopping, Fig. 2.14. In fact we are able to identify the 5 phases of the model from the $k$-means values assigned by the clustering algorithm and the phase transition lines by looking at the individual silhouettes coefficients of each point. Apparently, the distinctive characteristics of the points of this model allows us to distinguish the phases easily.

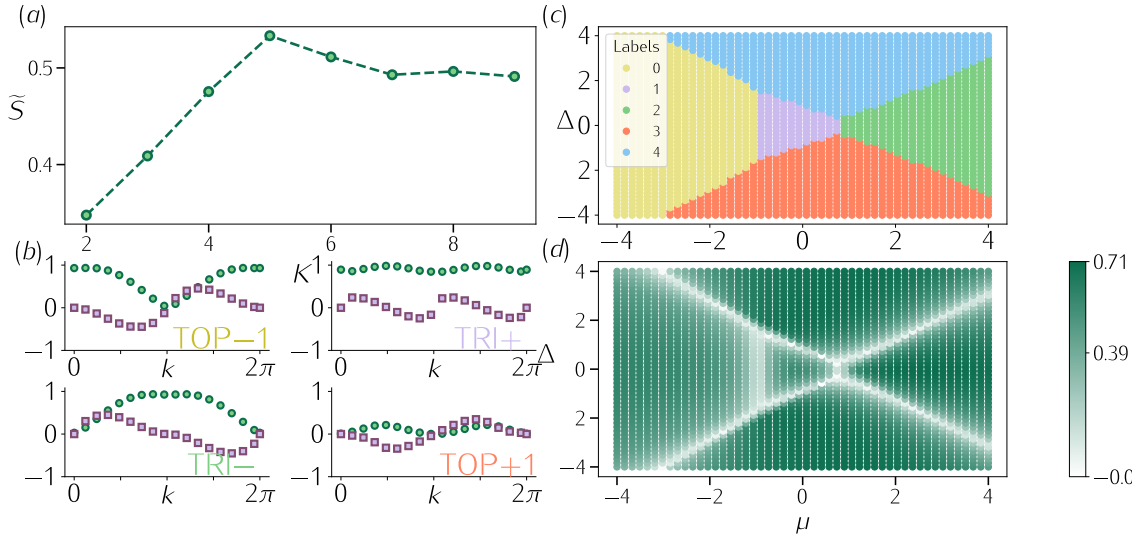



**Figure 2.14: $k$-means on NNN Kitaev.** (a) Average Silhouette coefficients for different cluster numbers $K$ suggesting exactly 5 clusters as predicted for this model. (b) The first four centroids of $K$-means with $K = 5$. (c) $K$-means labels assigned to each point correctly identifying the phases. (d) Silhouette coefficients of the points.

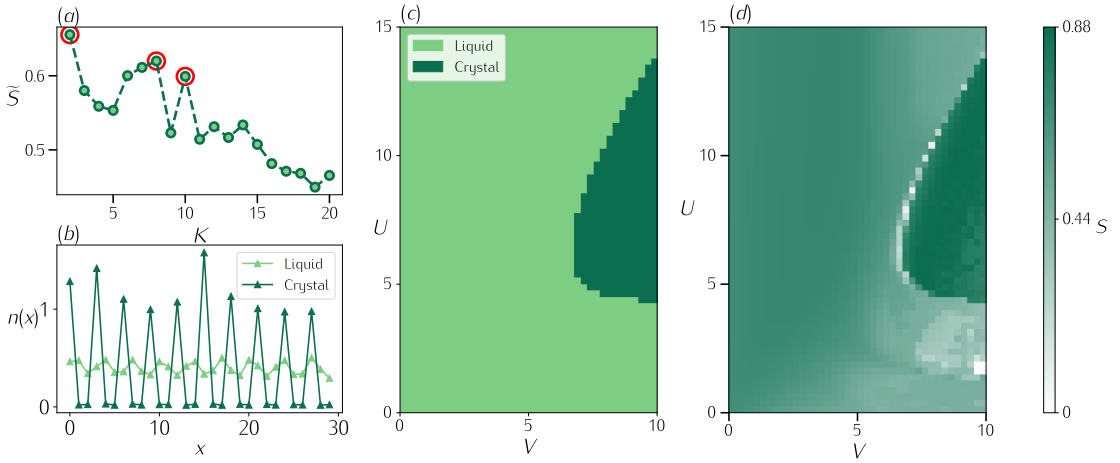



**Figure 2.15: Results of $k$-means algorithm for $K = 2$** . (a) Here plotted the silhouette score as a function of the number of clusters. Two evident peaks emerge for $n = 2$ and $n = 8$ clusters, here we show the results for $K = 2$. (b) Density part (last 30 entries) of the centroids. The crystallized phase presents far more localized particles. (c) Labels assigned by the algorithm. Clearly, the region $5 \lesssim U \lesssim 3V/2$ of the phase diagram appears to be radically different from the others. (d) The silhouette coefficients.

### 2.2.4 Extended Hubbard

We applied the algorithm to the design matrix $X_H$ we previously defined in Eq. 2.3. The average Silhouette score as a function of the number of clusters is plotted in Fig. 2.15(a). Two peaks clearly emerge at $K = 2$ and $K = 8$, with average Silhouette score $\tilde{S} = 0.65$ and $\tilde{S} = 0.62$ respectively. Since we expect a richer phase diagram compared to the previous model, these are the clusterizations of interest that we will consider and their results are plotted in Fig. 2.15 and 2.16 respectively. As it is customary we plot the labels assigned to the points after classifying them with $K$-means and the Silhouette score of each point. As suggested in [TMVE23] low Silhouette score can be considered as an indicator of a possible phase transition.

Clusterization with $K = 2$ (Fig. 2.15(c)) clearly resembles results of PCA, dividing the $5 \lesssim U \lesssim 3V/2$ region from the rest of the phase diagram. The nature of this separation can be understood by looking at the local density part of the centroids plotted in panel (b) of Fig. 2.15. In fact, in the $5 \lesssim U \lesssim 3V/2$ part of the phase diagram, the ground state presents far more localized particles, resembling a sort of *crystalline* phase, while in the rest of the phase diagram they are essentially delocalized, as one would expect for a *liquid* phase. This confirms the liquid to crystal phase transition that we had predicted in Chapter 1 Sec. 1.4.

**The $K = 8$ phase diagram** The clusterization with $K = 8$ is far richer and gives more physical insights into a reliable reconstruction of the phase diagram. In Figure 2.16(a, b) we plot the $k$-means labels ad the silhouette coefficients which already give us a picture of the phase diagram. In order to understand the nature of each clusters, that is of each phase, it is useful to examine the centroids by dividing them in their spin/charge structure part (Fig. 2.16(c)), and the local density part of the

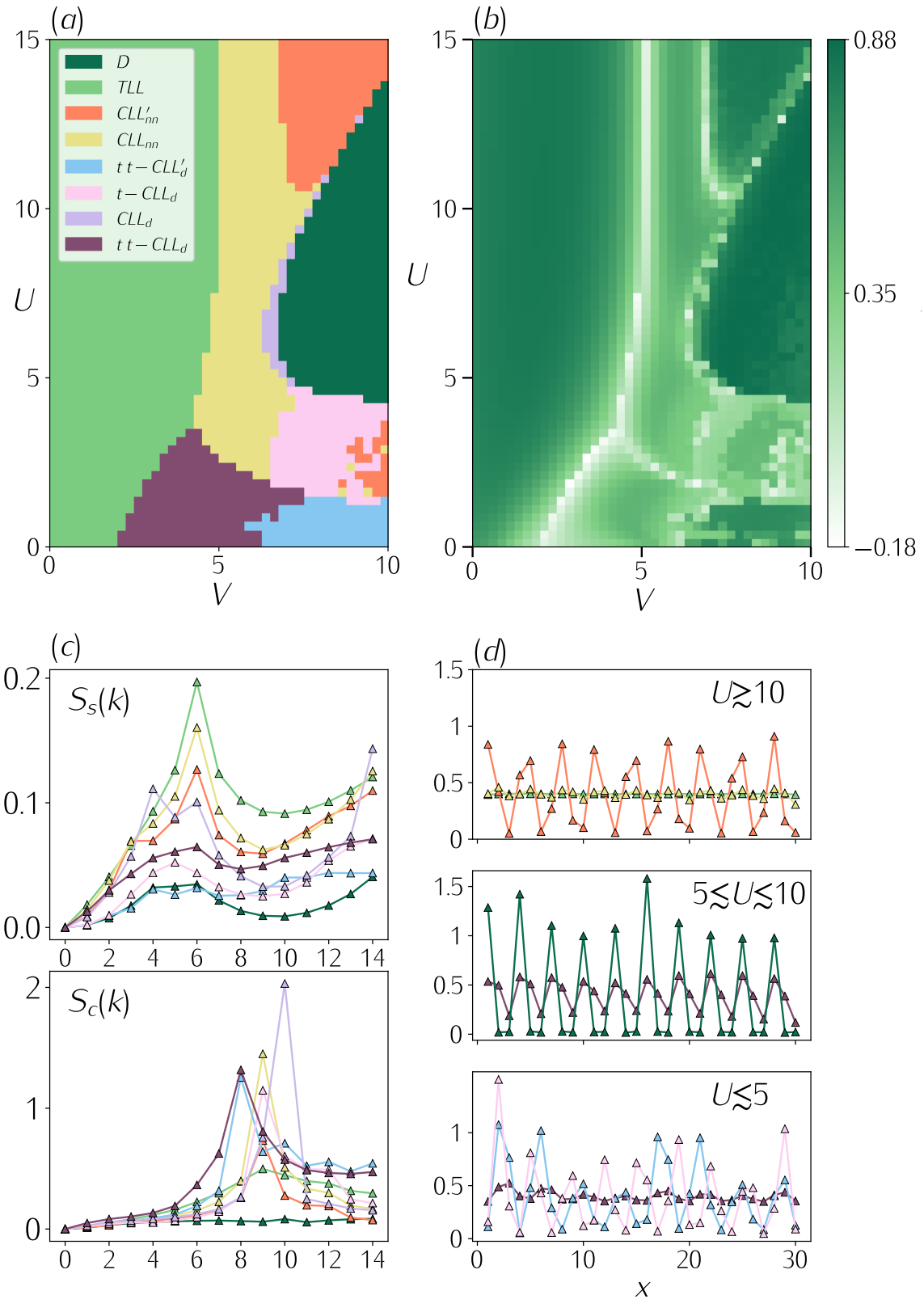



**Figure 2.16: Results of $k$-means for $K = 8$** . Clusterization in 8 subsets with $K$-means. (a) Labels assigned by $K$-means. (b) Silhouette coefficients. (c) Charge and Spin structure factor part of the centroids. More peaked density corresponds to higher particles localization. Phases with $M$ blocks shall have a peak in $S_c(k)$ at $k_c = 2\pi M/L$. (d) Local densities part of the centroids, plotted separated into categories according to their position on the phase diagram

centroids, which is plotted in Fig. 2.16(d).

We analyze the diagram from top to bottom. Starting from the high part of the phase diagram, i.e. $U \gg V$, there appears to be a separation into three phases, which we call, for growing values of $V$: $TLL$, $CLL_{nn}$ and $CLL'_{nn}$. Looking at the local density plot of these phases – represented in the upper panel of Fig 2.16(d) – one can see that, while $TLL$ and $CLL_{nn}$ (light-green and yellow) have almost perfect uniform density around 0.5, $CLL'_{nn}$ (orange) exhibits well-defined peaks, which correspond to more localized particles. On the other hand both $CLL_{nn}$ and $CLL'_{nn}$, contrarily to $TLL$, present a peak in the charge structure factor $S_c(k)$ for $k = 9\pi/15$, as plotted in the lower panel Fig 2.16(c). This, as reported in [MDLP13, DLC+15], corresponds to a cluster-ordering of particles. Transitions are identified, in this sector of the diagram for $U \gg V$, at $V \simeq 5.5$ and $V \simeq 7$. This findings agrees perfectly with our theoretical knowledge of the model in this sector cf. 1.6.

Moving to mid-range values of $U$ there is a transition between $CLL_{nn}/CLL'_{nn}$ to the previously described crystalline phase, which we will refer to as $D$ in the following and is shown in dark green in Fig 2.16(a). Furthermore, for $5 \simeq U \simeq 10$ and $V \simeq 7$, there appears to be a tiny cluster which divides $CLL_{nn}$ and $D$ highlighted in lilly. This is the same region that was highlighted by the seventh principal component (cf. Fig. 2.8). This tiny phase exhibits a well-pronounced peak in the charge structure factor for $k = 10\pi/15$, signalling again an underlying block structure, while local density is more compatible with liquidity w.r.t. the $D$ phase. In this sector we recollect the classical phase transition at $U$ $3/2V$. Yet, given the mid-range values of $U$ and $V$ we would expect the quantum fluctuations to lead from a cluster $CLL_{nn}$ type phase to a doublon liquid. Instead we find that only small fraction of the dataset is a liquid of doublons, the lilly $CLL_d$ phase, whereas the phase $D$ is crystallized.

Finally for small values of $U$ we see the emergence of three other phases. The first one, which we call $tt - CLL_d$, is found for $V \lesssim 7.5$ and has, again, uniform density, and charge structure factor peaked at $k = 8\pi/15$. As the off-site interaction becomes stronger the local density becomes more peaked. For $U \lesssim 2$ the charge structure is not modified, while for $2 \lesssim U \lesssim 5$ the peak shifts to $k = 9\pi/15$. We will refer to these phases as $tt - CLL'_d$ and $t - CLL_d$ respectively. On a final note, the third peak in Fig. 2.9(a) suggests a third classification with $K = 10$ (that we run and not show here for brevity) in which another phase arises next to $t - CLL_d$ for $V \gtrsim 7$ that can also be seen in the silhouette coefficients of Fig. 2.9(b).

The name of the last three phases suggests that we expected effects of quantum tunneling become relevant in this sector of the phase diagram. Further analysis that consider also a doublon and a nearest neighbour order parameters is presented in the soon-to-be-published work [CTEP24] in which we explore further the connection between the predictions of machine learning with the standard many body investigation of the phase diagram. The general picture depicted by $k$–means clustering will also be confirmed by other clustering algorithms such as t-SNE, which we analyze in the following section. Then, in order to benchmark these results we will exploit another unsupervised machine learning technique that was developed specifically for detecting phase transitions, i.e. Learning by Confusion.

## 2.3 t-distributed Stochastic Neighbor Embedding

t-distributed stochastic neighbor embedding [vdMH08] is a popular dimensionality reduction technique that creates a low-dimensional distribution of data which is faithful to the original one and thus helps visualizing clusters of data in 2 or 3 dimensions. It starts by creating a probability distribution from the Euclidean distances between data points in their original space. In particular given two points $x_i, x_j \in \mathcal{D}$ where $\mathcal{D}$ is the dataset, we interpret the conditional probability $p_{i|j}$ as a measure of similarity between the two points under a Gaussian centered at $x_i$:

$$p_{j|i} = \frac{\exp(||x_i - x_j||^2/2\sigma^2)}{\sum_{k\neq i}\exp(||x_i - x_k||^2/2\sigma^2)} \tag{2.11}$$

where $\sigma$ is the Gaussian band-width parameter that we will discuss later. t-SNE creates a symmetric joint probability $P$ from 2.11 by taking $p_{ij} = (p_{i|j} + p_{j|i})/2$. Then, it gives to each point $x_i$ a new set of coordinates in a lower-dimensional space $y_i \in \mathbb{R}^d$ for $d = 2$ or 3, where the similarity between points is given by a $t$-Student distribution $Q$ in the form:

$$q_{ij} = \frac{||y_i - y_j||^2}{\sum_{k\neq l}||y_k - y_l||^2} \tag{2.12}$$

In order to make $q_{ij}$ a faithful low-dimensional representation of the distribution $p_{ij}$, t-SNE minimizes the Kullback-Leibler divergence

$$C = KL(P||Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{2.13}$$

Typically this is done updating the position of points $y_i$ by gradient descent:

$$y_i = y_i - \eta \frac{\partial C}{\partial y_i} \tag{2.14}$$

with $\eta$ the learning rate. The parameter $\sigma$ appearing in 2.11 is chosen by the algorithm according to another hyperparameter called *perplexity* which needs to be set when running the t-SNE. Perplexity is defined as the exponential of the entropy of the distribution and can be interpreted as the average number of neighbors we expect the points to have in the original space and typical values range from 5 to 50 [vdMH08].

Setting the hyperparameters in this context is not easy because t-SNE does not offer a quantitative analysis of the goodness of clustering. It merely shows the data in a new reduce dimensional space and the rest is left for out interpretation. For this reason we are forced to select the hyperparameters by varying them in a relevant range and choose the division in clusters that seems to fit best to the data. In the results below we will show that t-SNE does not seem to be affected very much by changes in the two main hyperparameters: perplexity and early exaggeration. The

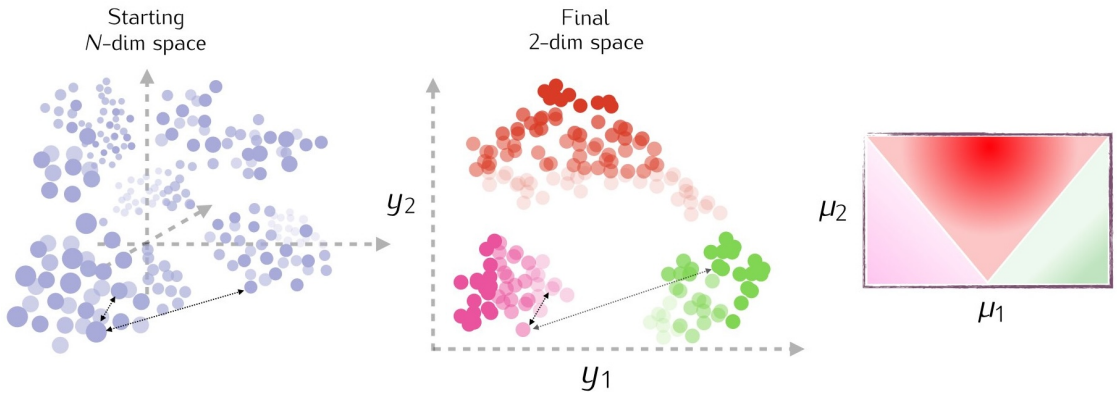



**Figure 2.17: t-SNE**. The datapoint live in a $N - dim$ space (left). The algorithm assigns a new set of coordinates $(y_1, y_2$ for each qubit in a reduced space of 2 dimensions in our case (center). Each point is colored differently so that we can plot it back in the phase diagram (right) according to its parameters $\mu_1, \mu_2$ and highlight the phase transition (s) if detected from the algorithm.

first one has already been explained whereas the latter is a parameter that enforces distance between cluster of points.

As an example, in Figure 2.17(b) we show a possible data encoding into a 2D space where the points are separated well-enough to allow us to assign them to different phases. Since we cannot assume this to be the case for every model we also color each point uniquely according to its embedded coordinates $(y_1, y_2)$. The same color is then used for the original point, shown in the phase diagram (c). This might allow for better detection of phase transitions.

### 2.3.1 Kitaev Chains

We ran t-SNE on our datasets varying both perplexity and learning rate $\eta$ without seeing major changes to the final result. Also, no relevant distinction was found in either reducing the dimensions to 3 or 2, so we stick to 2D.

In Figure 2.18 we plot the results of t-SNE on the non-interacting/interacting dataset. For both models we plot the dimensional reduction to 2 dimensions. In the non-interacting case we see four well-separated clusters and each one corresponds to one of the phases of the model. In particular t-SNE separates the two trivial phases (TRI+, TRI-) of the non interacting model in the same way as $k$-means does. For the interacting case the clusters are not as well-separated except for the CAT phase. Although the shape of the of the low-dimensional representation is not meaningful in t-SNE cluster, we are interested in finding the position of the points close to the phase transition lines. For this reason we adjust the brightness of each data point according to its silhouette value that was calculated in section 2.2.B (compare with Figs. 2.11 and 2.12). In this way we are able to see that the points closer to a phase transition are close to the margin of the cluster.

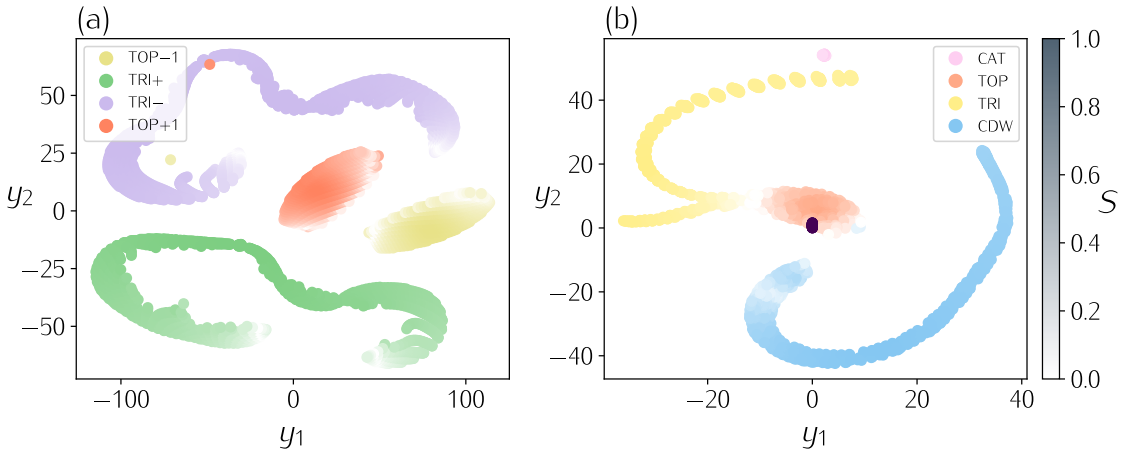



**Figure 2.18: Clustering with t-SNE on Kitaev data**. The plot shows the visualization of the data projected in 2D applying t-SNE (components $y_1, y_2$). In both panels we assign a color to each point corresponding to its phase in the model while the brightness of each point corresponds to its silhouette (see Section 2.2.B). (a) Projection of the non-interacting model data points. Four clusters appear evidently and they correspond to the 3 phases of the non-interacting model with the two trivial phases (TRI+, TRI-) separated. The silhouette of the points (brightness of the color) highlights the points close to a phase transition, which happen to be at the borders of each cluster. (b) Projection of the interacting model data points. The separation of the clusters is less neat compared to (a) but thanks to the silhouette we can see the borders of each cluster.

### 2.3.2 Kitaev NNN

The previous sections showed us that the extended Kitaev model has neat and easy to spot phases and we expect t-SNE to work efficiently in the separation task. This is indeed the case because as we can see in the graph of Figure 2.19 the new set of coordinates associated to each data point from the algorithm allows us to distinguish 5 well-separated sets of points. Each group of points corresponds indeed to one of the phases with winding number $\nu = 0, \pm 1, \pm 2$, confirming the power of t–SNE at clustering the models. The plot was obtained with perplexity 100 and early exaggeration 50. The values were selected after by swiping both hyperparameters in a range selecting the best clustering. As noticed with previous datasets there was no relevant change with different hyperparameters.

### 2.3.3 Extended Hubbard

The tricky–to–reconstruct phase diagram of the Extended Hubbard Hamiltonian we considered tackled also with t–SNE in a pretty efficient way. Given the large number of phases of this model we have to be a bit creative with the interpretation of the output of the algorithm. This is because t–SNE does not try to divide the points into cluster but merely to conserve their high-dimensional geometrical properties when projecting to 2 or 3 dimensions. For this reason, some clusters appear spontaneously while others are less likely to be spotted.

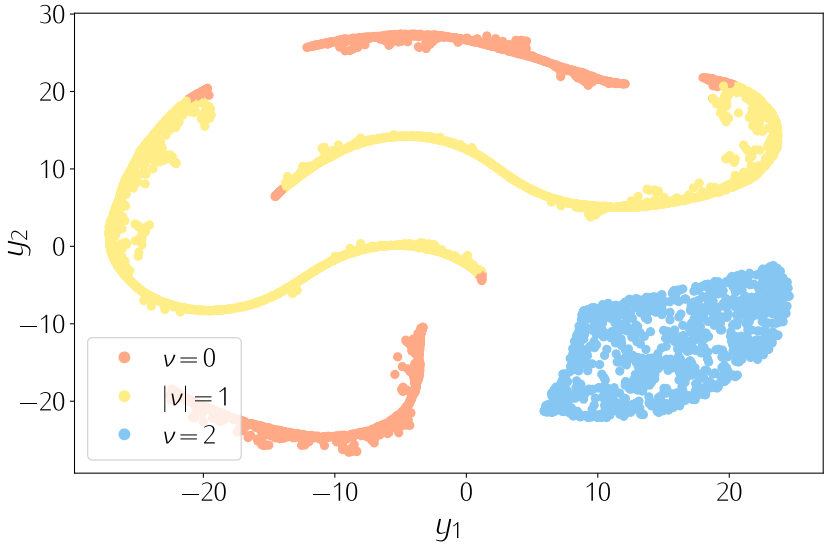



**Figure 2.19: t-SNE applied to Kitaev NNN**. The algorithm neatly separates the dataset in 5 clusters corresponding to the 5 phases of the model. Since the dataset is neatly separated we do not need further analysis so we colored each cluster with the corresponding label known a priori. Notice how a few points are misclassified and we checked by direct inspection that they lie on the phase transition lines. For this reason their mis-classification is due merely to discreteness of the dataset. The results were obtained with early exaggeration 50 and perplexity 100.

It is indeed our case, shown in Figure 2.20. The panel on the left shows the new set of coordinates $(y_1, y_2)$ assigned to each of the 4000 points of the dataset. We can clearly identify a few separated clusters, highlighted by numbers, that represent phases that have already been found by, e.g. $k$-means algorithm. Cluster 5, instead, encompasses 4 phases, in particular: $TLL$, $CLL_{NN}$, $CLL_d$, $tt-CLL_d$. This information is pretty trivial and would not allow us to recover many phases. For this reason in the left panel (b) we plot each point of the phase diagram color-coded with the coordinates of its t–SNE representation. This allows us to clearly separate almost all the four phases, with the transition between $TLL$ and $tt - CLL_d$ being not so evident. The last panel in the figure helps to figure out the color-coding scheme.

We acknowledge that we could have used a different metric to assign a label to each point, for example $\sqrt{y_1^2 + y_2^2}$ or $|y_1| + |y_2|$ might seem more intuitive. Yet, it would also compromise the geometric distribution realized by t–SNE. For example the first metric would make points in the upper left corner of (a) and the lower right placed close to each other, which we think would destroy the information gained by t–SNE and the meaning of this algorithm.

The choice for the hyperparameters used are early exaggeration 50 and perplexity 100. They were selected after a few trials in which we tried to explore the meaning of these parameters. Perplexity is proportional to the number of nearest neighbours considered by the algorithm and we immediately notice that the final KL divergence (the error of the algorithm) would increase with the perplexity. This would suggest that the higher the perplexity the better but trying with an excessive value of 1000

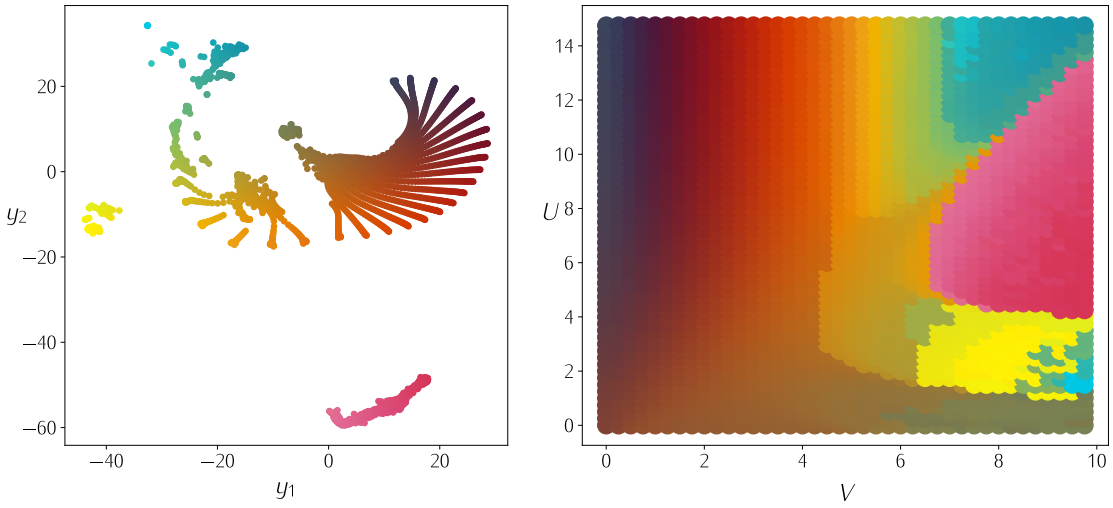



**Figure 2.20: t–SNE applied to Extended Hubbard** (a) Points of the model plotted in the embedded space with the new coordinates $(y_1, y_2)$. The position of the points produces a few evident clusters of different shape. To highlight the intra-cluster differences the scatter plot is color-coded according to the positions of the points. (b) Each point of the dataset is plotted with the same color of its corresponding embedded point. The results were obtained by using early exaggeration 50 and perplexity 100.

(which accounts 3000 nearest neighbours for every point, cf our dataset is 4000 points) we only get one big cluster of points that are not separated at all. That means that the final KL might not be a suitable parameter to check. For this reason we settled on a value of perplexity 100.

The second hyperparameter is early exaggeration. We varied it range $[10, 80]$ with steps of 10 because t–SNE is pretty fast at converging and we did not see interesting changes in the final cluster forms, so we settled on ee $= 50$.

## 2.4 Learning by Confusion

Learning by Confusion is a widely-known unsupervised method to determine phase transition points [DAR+22, RLKM22, vNLH17a]. The central idea is to use a supervised algorithm, in this case a Convolutional Neural Networks (CNN), to determine the phase transition point of a quantum system by scrambling the dataset until a good performance is reached.

More specifically, in the context of condensed matter physics we have a set of data that typically depends on, say, two parameters $\mu_1, \mu_2$ that span a whole phase diagram. To apply Learning by Confusion in its original form, we fix one of the two parameters $\mu_1$ and swipe the second one along a line of values $\mu_2 \in [A, B]$ generating a dataset of points that we assume might undergo one phase transition as shown in Fig. 2.21.

At each value of $\mu_2$ we train the CNN. To do so, every element of the dataset is

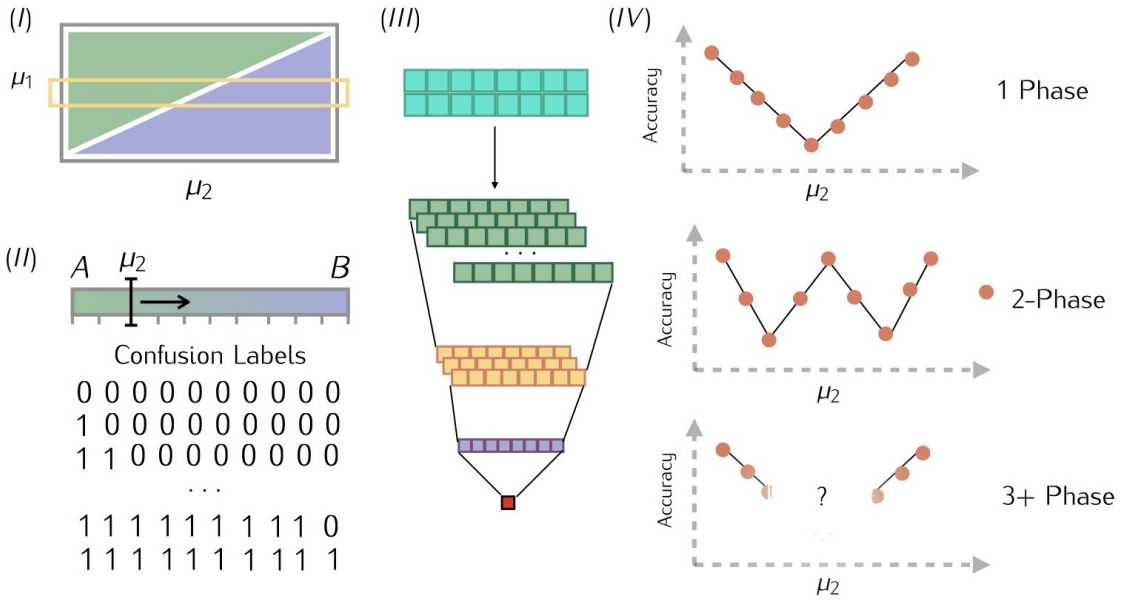



**Figure 2.21: Confusion Learning**. (I) We start by selecting a line of the phase diagram that may or may not cross a phase transition by fixing one parameter and changing the other one ($\mu_2$ in this case). (II) By swiping $\mu_2$ in the discretized interval $[A, B]$ we generate different labellings for our data going from all zeros to all ones. (III) Schematic of the convolutional neural network used in the process. Blue is the input data, in green, yellow and purple the intermediate layers and finally the accuracy is the red square representing the output neuron. For each labelling we train a convolutional neural network and plot its accuracy (IV). We expect the canonical $V$-shape and $W$-shape in the case of 0 (1) phase transition. There is no case in literature for multiple phase transitions.

assigned a label (0 and 1 in the case of two phases). The CNN is then trained in order to learn how to assign labels correctly. The degree of precision achieved by the network is evaluated by computing the *accuracy*, which is defined as the ratio between the number of right guesses and the total number of guesses, over a test set.

Now, the core of this technique is that: we initially label the dataset uniformly, e.g. assign to each point the label "0" corresponding to one of either phases. The CNN is subsequently trained and tested. As all points are assigned the same label the network learns easily to associate the label "0" to every input, resulting in a perfect accuracy. Once that has been done, the dataset is relabelled: the first element of the dataset (the one obtained by setting e.g. $\mu_2 = A$) is now assigned the label "1", while the rest of the data points remain "0". The CNN is then trained and tested again, and the accuracy is expected to decrease because we are forcing the CNN to classify the dataset in a wrong way. We then proceed to relabel the data by assigning the first two data points to "1" and the following ones to "0". This steps represent the confusion part of the algorithm because we are deliberately mislabelling our dataset. The process is repeated until uniform "1" labelling, and therefore again perfect accuracy is reached. We call this process the 2–phases

Learning by Confusion.

If the system under study undergoes a phase transition in the phase diagram region swept by the dataset, at a certain point during the confusion process the data will be correctly labelled according to the two phases. If that is the case, we expect a peak in the accuracy because the dataset is now labelled in a sensible way that the CNN can understand. This results in the so-called W-shape of the accuracy plot [vNLH17a]. In case of no phase transition we expect instead a $V$-shaped plot which is due to the deterioration of accuracy with the mislabeling. It is instead not clear what to expect in the case of two or more phase transitions. We will purposely investigate phase diagram lines that undergo 2 phase transitions to explore these effects. We applied this technique only on the more challenging datasets, that is the interacting Kitaev chain and extended Hubbard.

### 2.4.1 Extended Hubbard

For this model, we construct the matrix data points to feed to the CNN with the charge and spin structure factors and the local density arranged to form a $2 \times 30$ matrix

$$M_i = \begin{pmatrix} S_c^{(i)}(k_0) & \dots & S_s^{(i)}(k_0) & \dots \\ n^{(i)}(x_0) & \dots & n^{(i)}(x_{15}) & \dots \end{pmatrix}$$

where the index $i$ refers to a point in the phase diagram.

We apply 2–phases learning by confusion for two different lines in the phase diagram. In the first case we generate 1000 points with $U = 20$ fixed and sweeping $V \in [0, 10]$. In the second one we consider 1000 points with $V = 7.5$ and $U \in [0, 10]$. We always considered equally spaced values of $V$ and $U$ respectively. The results of 2-phase learning are summed up in Figure 2.22(b,c). The accuracy as a function of the transition point $V$ for the former case is plotted in panel (b) and for the latter in panel(c).

The fact that it does not show the previously cited W-shape hints at the fact that the division in two phases might not completely describe our dataset. This is in fact not surprising, as both theory and $K$-means algorithm predict three different phases for this particular line in the phase diagram (namely $TLL$, $CLL_{nn}$ and $CLL'_{nn}$, as depicted in panel (a)).

### 2.4.2 3-phase learning

To explore further the multiple-phases transitions that appear in our phase diagram we push the limits of the learning by confusion further. Here, we introduce a 3 phases generalization of Learning by Confusion. This is a natural extension of the method explained above but using 3 labels instead of 2, e.g."0", "1" and "2", and iteratively moving the two transitions points. Following this generalization idea we expect the diagram to have peaks of high accuracy at the corners (where the confusion dataset is always uniformly mislabelled) and one peak in middle. The coordinates of this maximum, e.g. $(V_1, V_2)$ are the two phase transition points.

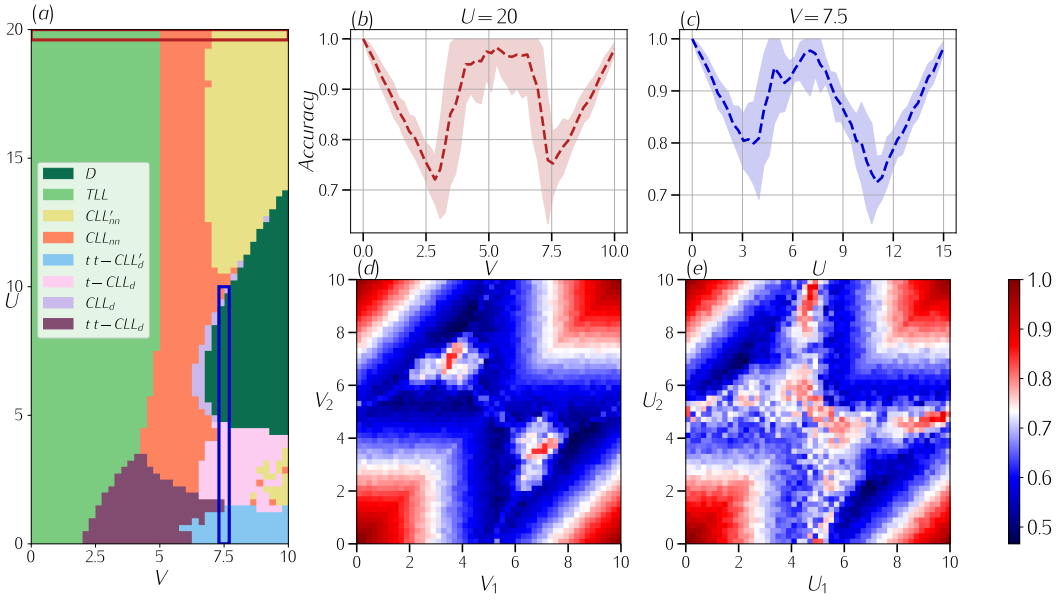



**Figure 2.22: Confusion learning of the Hubbard model**. (a) The model's phase diagram reconstructed with $k$-means (cf. 2.16) Red line: we set $U = 20$ and swiped $V \in [0, 10]$, results are shown in (b) for the 2-phase learning and (c) for the 3-phase learning. Blue line: we set $V = 7.5$ and $U \in [0, 10]$ and the results for 2-phase learning are in (d) and in (e) for the 3-phase learning.

**$U = 20$ phase transition** This is indeed what we find for the high $U = 20$, plotted in panel (d) of Fig. 2.22. Each column represents a different value for the first transition, that we call $V_c^{(1)}$, and each row a different value for the second one, called $V_c^{(2)}$. Results are clearly symmetrical w.r.t the diagonal since the transposition operation simply corresponds to an inversion between "1" and "2" labels. As expected, we observe a sort of $3d$ version of the W-shape, with the angles corresponding to uniform labelling corresponding to perfect accuracy of the network and a central peak indicating again the presence of three phases for this line of the phase diagram. In particular the peak of the accuracy, i.e. 98%, corresponds to $V_1 = 3.8$ and $V_2 = 7.2$. The results are averaged over 50 runs in order to decrease the variation due to the training of the neural network. The peak forming at the center stands out, this signals that the data is clearly separable into 3 different phases, the $TLL$, $CLL_{nn}$ and $CLL'_{nn}$, as we would expect from previous analysis.

**$V = 7.5$ transition line** The second line we considered is at $V = 7.5$ fixed case with $V = 7.5$ and $U \in [0, 10]$. In this case the plot of the accuracy for the 2-phases Learning by Confusion, represented in Fig. 2.22(e), is more compatible with the desired W-shape, with a peak emerging at $U_c = 4.7$. That being said, the accuracy still presents anomalous peaks, which suggest, together with predictions of $K$-means clustering, to apply again the 3–phases generalization of the algorithm. Results are plotted in Fig. 2.22. This is more challenging classification task and we can see that for many reason. Firstly, one can see that, differently w.r.t Fig. 2.22(d), there are new peaks in the middle of the "borders". These are in agreement with the previously stated compatibility of 2–phases accuracy plot with the W-shape. Consider in fact,

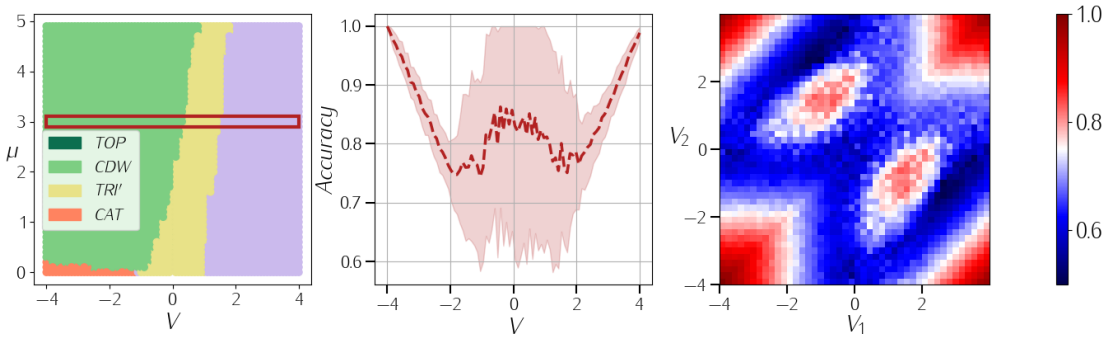



**Figure 2.23: Confusion Learning of the Kitaev interacting model**. (a) The model's phase diagram with the red line at $\mu = 3$ while $V \in [-4, 4]$. Results are shown in (b) for the 2-phase learning and (c) for the 3-phase learning.

for example, the first row; this corresponds to placing the first transition at $U = 0$. Therefore, the peak in the middle of the lower border corresponds to a 2–phases system with phase transition at $U_c = 4.7$. This happens identically on the last column (with $U_1 = 10$), and on the diagonal $U_1 = U_2 \sim 4.7$, as expected. In addition to these, a new non-trivial maximum of the accuracy emerges for $U_c^{(1)} = 2.2$ and $U_c^{(2)} = 4.6$. This last prediction is in perfect agreement with $K$-means clustering.

### 2.4.3 Kitaev Chain

To confirm the validity of our extended learning by confusion scheme we also apply it to the interacting Kitaev chain. If we consider a line of points at $\mu = 3$ fixed varying $V \in [0, 8]$ we cross two phase transitions from trivial (TRI) to topological (TOP) to charge density wave (CDW) as highlighted in Figure 2.23(a).

We divided a set of 880 points in 44 sections and ran the 2-phase learning method (b) we again obtain a very unclear shape of the accuracy in the central part of the graph where we expect more than one phase transition. The standard deviation in particular being very large makes us assume that it is needed to use the 3-phase learning scheme, whose results are plotted in the next panel (c) and confirm the expected 3d version of the W-shape graph. In particular, there is a peak of accuracy at around $(-0.6 \pm 0.2, 1.3 \pm 0.2)$ which agrees with values that we have from the theory (Sec. 1.4) but also with applying the various machine learning models of this chapter.

It is also important to consider that the precision of this method relies in the number of partitions of the dataset. In this case having 880 points divided in 44 sections we have an uncertainty on the result of $8/44 \sim 0.2$ so we still cannot retain a precise phase transition point. As a final remark, there is a chance that some generalization beyond 3-phase learning might actually find the third phase transition, that is the transition inside the CDW sector that up to now we have ignored for simplicity.

# Conclusions

In this part of the work we have shown how to develop techniques which combine machine learning to predict the topological and non-topological quantum phases of paradigmatic models. In particular there are a few main points on which we want to concentrate.

**Transfer learning** A first interesting result is the ability of algorithms (as simple as principal components analysis) used to learn the phase diagram of a model to transfer this information into a new model, more complicated than the previous one. This was the case for the PCA applied to the Kitaev chain which not only successfully highlights all the phases of the model but with the same information from the datapoints can also classify perfectly the interacting system. The same, unexpected, result was obtained with $k$-means which is not conceived to be used for transfer learning but yielded satisfactory results towards this application. t-SNE and Learning by Confusion by definition do not allow us to be used from a model to another and cannot be taken into consideration for this analysis.

**Interpretability** A second interesting point is the interpretability of some of these techniques. PCA and $k$-means in particular offer many clues to what the algorithm understands from the data. PCA explained variances give a quantitative measure of how much the data can be compressed into way less feature that the original ones, often the number of the most relevant ones (i.e. largest) is related also to the number of phases. Along with that, the shape of the eigenvectors is a clear indicators of the differences that characterize the data and why they can be separated. Finally, the explained variance puts this information together giving us a clear picture of the model we are exploring.

$k$-means allows also for interpretability. First of all it allows us to find a priori the number of phases by looking at the silhouette score for different number of clusters $K$. Secondly, the silhouette coefficient of each point is of utmost importance in detecting the points that lie on a phase transition as it was extensively noticed in each of the 6 models considered in this work. This information is more relevant than the simple classification done automatically obtaining the labels from the $K$-means algorithm because it can tell us many details on the structure of the phase diagram, for example the CDW and I-CDW of the Kitaev Interacting model can be seen easily thanks to this method, the same goes for all the small phases at large $V$ and small $U$ of the Extended Hubbard model we considered. Finally, the centroids – much as for the eigenvectors of PCA – tell us why the algorithm chooses such classifications

and were used in concomitance with the silhouette coefficients to understand the physical properties of the different phases, e.g., in the Extended Hubbard model.

Clearly, these considerations do not apply to t-SNE despite it being a powerful algorithm, since it can merely embed points into a lower dimensional space. Yet, as we saw with each model the clusterization can either be very evident like with Kitaev NNN or Kitaev non-interacting, while being less trivial for Kitaev Interacting and Extended Hubbard. This is shown by the geometry of the points in the embedded space that allowed us to create a shadow of the phase diagram in line with the predictions of the other models.

Learning by confusion, contrarily to other three was invented with the purpose of having a classification of data in condensed matter that comes from a reasoning similar to human intuition. This is what allowed us to use it also beyond its original intended scope inventing the 3-phase learning.

**Systematic approach** This last point inspires our final but not least important remark. We believe that we showed systematically that it is possible to have a conscious approach in the study of phase diagrams which combine more traditional approaches from quantum many-body theory and machine learning. Throughout this part we used the machine learning with a few precautions.

First of all, our complete agnostic approach to the models. In no part of the algorithm we assumed we knew what the phase diagram of each model would look like, there are in fact two notable examples that confirm this. A simple one, the CAT-like phase in the interacting Kitaev chain was not known to us (or had not been considered) and it was found because it emerged explicitly from PCA and $k$-means analysis. A more sophisticated one, the whole analysis of the Hubbard model produced a plethora of phases that were not considered a priori or known in literature and were later explored with more traditional numerical techniques to show their existence, the power of this methods in fact required a follow-up work that is currently in preparation [CTEP24].

Following the agnostic reasoning, which is the reason why the algorithms are unsupervised, we did not simply applied the algorithms but for each one tried to create a pipeline so that they could be used in the same way for many other models.

Finally, we pushed the interpretation of this methods to its limits, for example with the learning by confusion scheme, which lead us to invent an extension of this technique, useful for multi-phase transition sectors of the phase diagram.

As it is clear from this work we did not consider scaling the size of the systems to probe the success of these algorithms in the context of the therdmodynamic limit. This was not done mainly because it would require a lot of time to generate that many datasets, in particular for complicated models like the Extended Hubbard, and was not considered at the time of the work. Although this remains an interesting question to open, we did try, at least in the case of the Kitaev model, to generate larger instances of data for example with $L = 200$ and noticed very few numerical differences in the correlation functions. This suggests in a qualitative way that the algorithms might be effective even for larger system sizes.

Putting all together, we might confidently say that all the results summed up above guarantees one last aspect of this research: reproducibility. Being able to classify phases not only to an exemplary model like the Kitaev chain but even the Extended Hubbard gives us confidence that our methods can be used in any other topological model of interest.

# Part II

# Quantum for Classical

# Motivation: Optimization of Quantum Circuits on Neutral Atoms

Hybrid quantum-classical variational algorithms [MRBAG16, PMS+14, MBB+18] play a central role in the current research on noisy intermediate-scale quantum (NISQ) devices [Pre18]. In a hybrid variational setting, a classical computer is entrusted with the non-trivial task of optimizing the parameters of a quantum state. These algorithms implement a heuristic protocol to approximately solve variational problems including combinatorial optimization tasks, which are ubiquitous and have a great practical importance [DP99], and are indeed one of the main drivers of the industry interest towards quantum computing applications. Unfortunately, the problems belonging to this class are hard to solve with classical methods [GJ82]. In this work we focus on the Max-Cut and the Max Independent Set (MIS) problems defined on specific graph instances.

MIS is part of a large set of combinatorial problems that include also Max-Clique and it was chosen because it is particularly suited for the class of quantum computer that uses neutral atoms. The idea to use neutral atoms for quantum computation was developed in the early 2000s [BCJ+00] and since then it has reached impressive results, with the current capability of simulating many-body states of hundreds of atoms [EWL+21], solving combinatorial problems on graphs [DHK+23], improving on quantum machine learning [HTDH21] and providing impressive results on quantum error correcting codes [BEG+23]. This type of computers exploit Rydberg blockade to entangle qubits. This phenomenon prevents electrons to be excited at the same time when two atoms are close enough. It turns out that this is a condition for the solution of many combinatorial problems and for this reason these problems are used for benchmark on the current NISQ algorithms like QAOA.

Among the hybrid variational algorithms, the Quantum Approximate Optimization Algorithm (QAOA) [FGG14] is in fact extensively studied [ZWC+20] as a promising algorithm to investigate quantum speedups on NISQ devices and has been implemented on several experimental platforms, such as the Rydberg atom arrays [EKC+22] we just cited, superconducting processors [HSN+21], trapped-ions simulators [PBB+20], as well as simulated on classical devices [MC21].

Similarly to other hybrid variational algorithms, QAOA consists in a sequence of parametrized quantum gates applied to a wavefunction, on which an expectation value of some operator, typically the Hamiltonian, is reconstructed from measurements. The task of the classical subroutine is to optimize the gate parameters

in order to minimize such expectation value. Every variational quantum algorithm therefore requires the estimation of the expectation value of a Hamiltonian [HKP20] for which the number of measurements scales with the dimension of the problem. Moreover, the interplay between the classical and quantum parts of the algorithm entails to run the quantum circuit a large number of times, thus being expensive in term of resources because not all the current proposal for quantum computers have an efficient repetition rate. In particular, the Rydberg atoms' platforms on which we focus in this work run with a repetition rate of $1 \sim 5$ Hz [HBS+20]. Finally, there is the notorious problem of Barren Plateaus (BP) which are large portions of the optimization landscape in which the gradient becomes exponentially small with the number of qubits and layers [MBS+18]. This phenomenon was proven to be caused also by the presence of noise [WKJC21] or by the use of a cost function depending on global observables [CSV+21].

To overcome these issues that emerge both in neutral atoms' platforms and other platforms, in fact, an efficient classical optimization routine is crucial. Different techniques have been proposed for optimizing variational quantum circuits, e.g. Nelder-Mead [GS17], Machine Learning [WHT16], gradient descent [WHJR18], iterative schemes [ZWC+20, MMS+22, MFS19], Gaussian Processes [SKS23, MLIdJ22] and Bayesian methods [O+17, ZLB+19, SKS+21, WKJC21, TY22]. In particular, to tackle the problem of BPs it might seem logical to avoid the calculation of the gradient. However, in [ACC+21] it was shown that gradient-free optimizers such as COBYLA, Powell and Nelder–Mead suffer from BPs too.

In this thesis work we want to present a Bayesian optimization framework developed especially to be ran on neutral atom quantum computers. Bayesian optimization is in fact extremely suitable for gradient-free global optimization of black-box functions [SSW+16, Fra18] and we present it in Chapter 3 without focusing on any particular type of platforms. We explore its behaviour in comparison with other global optimizers and we show that the convergence rate to a local minimum is faster. We demonstrate that the Bayesian approach is efficient in terms of number of circuit runs and is robust against noise sources. Then, in Chapter 4 we present the neutral atoms computer of the company PASQAL. We show the modifications needed to perform QAOA optimized with Bayesian Optimization on their computer and then the results of a few experiments run on the real machine. These first sections are developed from our published work [TVTE23].

We conclude this part, in Chapter 5 explaining and developing a different technique called Measurement Based Quantum Computation, how it can be used on Rydberg atoms and what resources are needed to perform it. This is part of an ongoing work that will be developed further also after the publication of this thesis.

# 3 | Bayesian Optimization for QAOA

## 3.1 QAOA for combinatorial problems

The QAOA is a variational quantum algorithm that performs hybrid quantum-classical optimization [FGG14]. This algorithm was introduced as an alternative to classical approximate methods to find the solution to hard combinatorial problems. The main advantage of using a quantum version of the algorithm is to exploit the search for the solution in the exponentially growing space of the qubits, with the action of relatively simple quantum algorithms. Given a cost function $C(z)$ with $z = (z_1, \ldots, z_i, \ldots, z_N)$ with $z_i \in \{0, 1\}$, QAOA aims at finding the bitstring $z^\star$ that minimizes the cost. In order to do so, the cost function is translated into a quantum operator $H_C$. This is done by replacing each binary variable $z_i$ with a two-level quantum state $|z_i\rangle$ and each $z_i$ term appearing in the cost function with a Pauli matrix $Z_i$. Since $Z_i$ is diagonal on the qubit $|z_i\rangle$, $H_C$ is diagonal in the computational basis $|z\rangle = |z_1 \ldots z_i \ldots z_N\rangle$ for $N$ qubits. This means that applying $H_C$ to $|z\rangle$ gives the classical cost $C(z)$ of such string, i.e.:

$$H_C |z\rangle = C(z) |z\rangle . \tag{3.1}$$

The QAOA circuit consists in preparing an initial state of $N$ qubits, usually $|+\rangle = \sum_z |z\rangle/\sqrt{2^N}$, and then applying two unitary operators alternatively: one generated by $H_C$, the other generated by $H_M = \sum_i X_i$, where $X_i$ is the flip (NOT) operator acting on the $i$-th qubit. The two unitaries together form one layer of the circuit and the operation is iterated for a number of layers $p$ which is called the depth of the circuit. The problems that we consider in this work (see Sec. 3.3) have a cost function at most quadratic in the binary variables. This means that $H_C$ comprehends only $Z_i$ and $Z_iZ_j$ terms. For this reason, we can implement $H_C$ by applying only rotations $e^{-itZ}$ on the qubits and gates $e^{-itZ_iZ_j}$ on the pairs of qubits. Putting all together, the QAOA circuit prepares the state

$$|\boldsymbol{\theta}\rangle = \prod_{l=1}^{p} e^{-i\beta_l H_M} e^{-i\gamma_l H_C} |+\rangle, \tag{3.2}$$

where $\boldsymbol{\theta} = (\boldsymbol{\gamma}, \boldsymbol{\beta})$ are $2p$ parameters. By measuring the state $|\boldsymbol{\theta}\rangle$ in the computational basis we obtain the probability amplitudes of each bitstring. In this way, by using relation (3.1) an estimate of the energy $E(\boldsymbol{\theta}) = \langle\boldsymbol{\theta}|H_C|\boldsymbol{\theta}\rangle$ is obtained. This energy is then fed to a classical routine which looks for the set of angles $\boldsymbol{\theta}^\star = (\boldsymbol{\gamma}^\star, \boldsymbol{\beta}^\star)$

that minimizes $E(\boldsymbol{\theta})$. Several strategies have been proposed for finding the optimal parameters $\boldsymbol{\theta}^\star$. In this work we rely on Bayesian optimization.

### 3.1.1 Combinatorial problems

*Max-Cut* – Given a graph $G = (V, E)$ where $V$ is the set of nodes and $E$ the set of edges, the Max-Cut problem consists in finding a partition of the graph's vertices $V$, $P = \{V_0, V_1\}$, such that the number of edges between $V_0$ and its complement $V_1$ is as large as possible. It is known to be a NP-hard problem [Edw73]. We can define the assignment of the nodes to the sets $V_0$ and $V_1$ by labelling with label "0" the nodes $v \in V_0$ and with label "1" the nodes $v \in V_1$. In these terms, the Max-Cut consists in finding the largest number of edges connecting the bits labelled with "0" to the bits labelled with "1". On a quantum computer, the labels 0 and 1 are replaced by the computational basis states $|0\rangle$ and $|1\rangle$ and the cost Hamiltonian can be written as:

$$H_C^{\mathrm{MC}} = -\sum_{(i,j)\in E} (1 - Z_i Z_j)/2. \tag{3.3}$$

The states with minimum energy then represent the bitstrings that maximizes the number of edges with two opposite values on their vertices.

*MIS* – The Max Independent Set problem consists in finding the largest number of graph nodes which are not adjacent. The corresponding cost Hamiltonian in its classical formulation [Luc14] is

$$C(x) = -\sum_{i\in V} x_i + \omega \sum_{(i,j)\in E} x_i x_j \tag{3.4}$$

where $x_i = 0, 1$, $\omega$ is a parameter that balances the effect of the first term (which maximizes the number of bits in $|1\rangle$) and the second one (which prevents neighbour bits to be activated at the same time). In order to translate the problem into its quantum version we make the variable substitution $x_i = (1-z_i)/2$ so that $z_i = +1, -1$. Then, we replace each $z_i$ with $Z_i$ and obtain the quantum Hamiltonian (discarding constant terms):

$$H_C^{\mathrm{MIS}} = \sum_i \frac{Z_i}{2} + \omega \sum_{(i,j)\in E} \frac{Z_i Z_j - Z_i - Z_j}{4}. \tag{3.5}$$

During the optimization process we monitor the approximation ratio $R = E(\boldsymbol{\theta})/E_{GS}$ [FGG14] where $E_{GS}$ is the energy of the solution bitstring. Since $E_{GS} < 0$ (due to our definition of the problems' Hamiltonians (3.3), (3.5)), $|E(\boldsymbol{\theta})| \leq |E_{GS}|$ and thus $R$ is upper bounded by 1. We also look at the fidelity defined as $F = |\langle \boldsymbol{\theta}|z^\star\rangle\langle z^\star|\boldsymbol{\theta}\rangle|^2$ where $|z^\star\rangle$ is the state which encodes the solution. The following results are obtained on two 3-regular graphs of 6 and 10 nodes which are plotted in the insets of Figs. 3.1(a)–(b).

## 3.2 Bayesian optimization

Bayesian optimization is a global optimization strategy which allows us to find within relatively few evaluations the minimum of a noisy, black-box objective func-

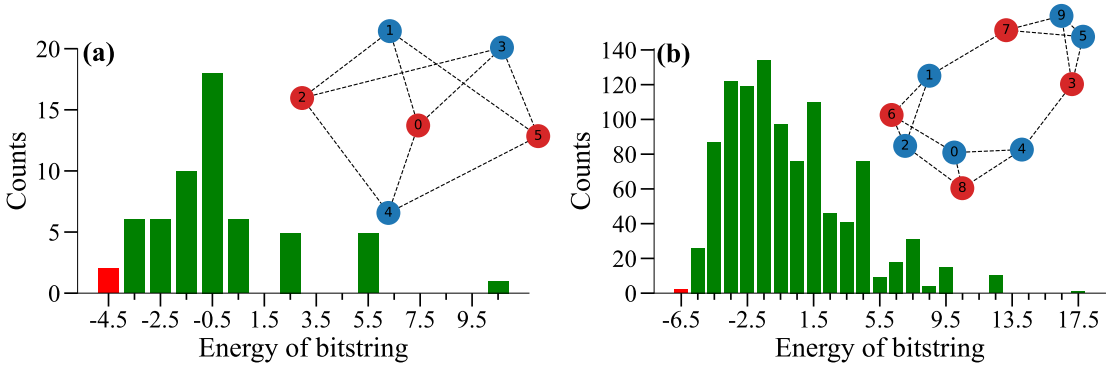



**Figure 3.1: Energy distributions of graphs.** (a) MIS: Distribution of the energies of the possible bitstrings for the graph of 6 nodes (shown in the inset, nodes in red correspond to one solution). The red bar to the left highlights the two solution bitstrings of the MIS problem on such graph. (b) Max-Cut: Distribution of the energies of the possible bitstrings for the graph of 10 nodes (shown in the inset, nodes in red correspond to one solution). The red bar highlights the two solution bitstrings of the Max-Cut problem on such graph.

tion $f(\boldsymbol{\theta})$ that is in general expensive to evaluate [SLA12]. The algorithm can be summarized as follows: (i) it treats the objective function $f$ as a random function by choosing a prior (also called surrogate model) for $f$. Several choices for the surrogate model are possible [SSW+16], in this work we adopt the so-called Gaussian process [RW05]. (ii) The prior is then updated through the likelihood function by gathering observations of $f$ and therefore forming the posterior distribution. (iii) The posterior distribution is finally used to construct an auxiliary function, called acquisition function, that is in general cheap to evaluate. The point where the acquisition function is maximized gives the next point where $f$ will be evaluated [Fra18]. See Appendix A.1 for an overview of Bayesian terminology.

Since Bayesian optimization requires no previous knowledge on $f$, it appears to be a well-suited technique for optimizing the parameters of a variational circuit running on NISQ devices. In the following sections we describe the Gaussian process, the optimization routine and the acquisition function in detail.

### 3.2.1 Gaussian process

Since the function $f(\boldsymbol{\theta})$ ($\boldsymbol{\theta} \in A \subset \mathbb{R}^d$) to be optimized is unknown, we may think of it as belonging to a random process, i.e. an infinite collection of random variables defined for every point $\boldsymbol{\theta} \in A$. A random process is called Gaussian if the joint distribution of any finite collection of those random variables is a multivariate normal distribution defined by a mean function $\mu(\boldsymbol{\theta})$ and covariance (or kernel) function $k(\boldsymbol{\theta}, \boldsymbol{\theta}')$ [RW05]. The mean function is the expected value of the function $f$, while the kernel estimates the deviations of the mean function from the value of $f$:

$$\mu(\boldsymbol{\theta}) = \mathbb{E}[f(\boldsymbol{\theta})], \tag{3.6}$$

$$k(\boldsymbol{\theta}, \boldsymbol{\theta}') = \mathbb{E}[(f(\boldsymbol{\theta}) - \mu(\boldsymbol{\theta}))(f(\boldsymbol{\theta}') - \mu(\boldsymbol{\theta}'))], \tag{3.7}$$

where $\mathbb{E}$ denotes the expectation w.r.t. the infinite collection of functions belonging to the random process. Conceptually, the mean encloses the knowledge of the function $f$ to reconstruct while $k$ represents the uncertainty we have on such reconstruction.

Since we assume $f$ to be smooth, we choose for $k$ the Matérn kernel, a stationary kernel [RW05] that depends on the distance between the points $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$, defined as

$$k(\boldsymbol{\theta}, \boldsymbol{\theta}') = \sigma^2 \left(1 + \frac{\sqrt{3}||\boldsymbol{\theta} - \boldsymbol{\theta}'||_2}{\ell}\right) e^{-\frac{\sqrt{3}||\boldsymbol{\theta}-\boldsymbol{\theta}'||_2}{\ell}} + \sigma_N^2 \mathbb{I} \tag{3.8}$$

where $||\cdot||_2$ is the 2-norm and $\sigma^2$, $\sigma_N^2$ and $\ell$ are three hyperparameters characterizing the Gaussian process. The hyperparameter $\sigma^2$ defines the variance of the random variables and $\ell$ is a characteristic length-scale which regulates the decay of the correlation between points: in the limit of $\ell \to \infty$ all points are equally correlated, for $\ell \to 0$ all points are uncorrelated. The noise parameter $\sigma_N^2$ is a value added to the diagonal accounting for random fluctuations around the true value of $f(\boldsymbol{\theta})$.

### 3.2.2 Bayesian optimization algorithm

The main steps of the algorithm for Bayesian optimization can be summarized in the pseudocode in Algorithm 1 (see also Appendix A.1.1 for details).

**Algorithm 1:** Pseudo-code for Bayesian optimization

```
Set the prior on f as a Gaussian process;
Evaluate f at N_W different points θ_i;
Define the initial training set D = {(θ_i, f(θ_i))}_{i=1}^{N_W};
Compute the hyperparameters σ², σ_N², ℓ based on D;
Set the guess for the minimum of f to f_m = min[{f(θ_i)}_{i=1}^{N_W}];
while n ≤ N_BAYES do
    Update the posterior distribution on f using the training set D;
    Compute the acquisition function with the updated posterior;
    Find θ̃ that maximizes the acquisition function;
    Evaluate f(θ̃);
    if f(θ̃) < f_m then
        Set the guess for the minimum of f to f_m = f(θ̃);
    end
    Append (θ̃, f(θ̃)) to D;
    Compute the new hyperparameters σ², ℓ;
    Increment n;
end
return f_m
```

The optimization starts by a warm-up phase where a number $N_W$ of evaluations of the objective function $f$ is performed. These evaluations take place at randomly chosen values of the points $\boldsymbol{\theta}_i$ and are collected in the training set $\mathcal{D} = \{(\boldsymbol{\theta}_i, y_i =$

$f(\theta_i))\}_{i=1}^{N_W}$ of the optimization. Given the set $\mathcal{D}$, we define the design matrix $\boldsymbol{\Theta} = (\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_{N_W})$ with the points and the vector $\boldsymbol{y} \in \mathbb{R}^{N_W}$ with the observations via $\boldsymbol{y} = (y_1, \dots, y_{N_W})$. We form the covariance matrix $\boldsymbol{K} \in \mathbb{R}^{N_W \times N_W}$ by evaluating the covariance function in Eq. (3.7) for each pair of points $\boldsymbol{\theta}_i, \boldsymbol{\theta}_j \in \boldsymbol{\Theta}$ via

$$\boldsymbol{K}_{i,j} = k(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j), \tag{3.9}$$

where $\boldsymbol{K}_{i,j}$ denotes the $(i, j)$ element of the matrix $\boldsymbol{K}$. The hyperparameters entering the kernel function (Eq.(3.8)) are optimized at this step, as explained in Sec. 3.2.4.

The training set will be used at each step of the optimization to incorporate the acquired knowledge in the Gaussian process. This happens in two steps. First, the Gaussian process prior is conditioned on the observations in $\mathcal{D}$ [RW05]. Conditioning is equivalent to a Bayesian step in which we multiply the prior with the likelihood, thus obtaining a posterior distribution (see Appendix A.1). Thanks to the properties of Gaussian distributions, the posterior is still described by a Gaussian process multinomial distribution but it is characterized by a posterior mean $\mu'$ and covariance $k'$ given by

$$\mu' = \boldsymbol{\kappa}^T \cdot (\boldsymbol{K} + \sigma_N^2 \mathbb{I})^{-1} \cdot \boldsymbol{y} \tag{3.10}$$

$$k' = k(\boldsymbol{\theta}, \boldsymbol{\theta}) - \boldsymbol{\kappa}^T \cdot (\boldsymbol{K} + \sigma_N^2 \mathbb{I})^{-1} \cdot \boldsymbol{\kappa} \tag{3.11}$$

Here, $\boldsymbol{\theta}$ is a generic point in $A$ and $\boldsymbol{\kappa}$ is a column vector formed by evaluating the covariance function $k$ between the generic point $\boldsymbol{\theta}$ and all the points in $\boldsymbol{\Theta}$, i.e. its $j$-th element is $\kappa_j = k(\boldsymbol{\theta}, \boldsymbol{\theta}_j)$. Eq. (3.10) shows that the new mean is a linear combination of the observations $\boldsymbol{y}$.

### 3.2.3 Acquisition Function

The next step in the Bayesian optimization is computing the acquisition function, whose maximum gives the next point at which to evaluate the objective function. A common choice of acquisition function is the Expected Improvement (EI): this function suggests which points, on average, improve on $f_m$ the most [Fra18]. This choice corresponds to defining the acquisition function $\mathrm{EI}(\boldsymbol{\theta}) = \mathbb{E}[u(\boldsymbol{\theta})]$ as the average value of the utility function $u(\boldsymbol{\theta}) = \max[0, f_m - f(\boldsymbol{\theta})]$ such that the lower $f(\boldsymbol{\theta})$ is with respect to the current minimum, the larger the utility $u(\boldsymbol{\theta})$ will be.

By considering that $f(\boldsymbol{\theta})$ is a Gaussian process, we can obtain an analytical expression for $\mathrm{EI}(\boldsymbol{\theta})$ as

$$\mathrm{EI}(\boldsymbol{\theta}) = \Phi(z)(f_m - \mu') + \phi(z)k', \tag{3.12}$$

where $\mu'$ and $k'$ are obtained for the point $\boldsymbol{\theta}$ by using Eqs. (3.10) and (3.11); $\Phi(\cdot)$ and $\phi(\cdot)$ are respectively the cumulative distribution function and the probability density function of the standard normal distribution and the quantity $z$ is defined as $z = (f_m - \mu')/k'$. The two terms in Eq. (3.12) represent the trade-off between exploitation and exploration: the first term, being proportional to the difference between the current minimum and the mean value of the posterior, brings the optimization towards points with lower $\mu'$ whereas the second one promotes points with larger $k'$, i.e. with higher uncertainty. The point $\tilde{\boldsymbol{\theta}}$ that maximizes the acquisition function is then added to

the training set $\mathcal{D}$ and the algorithm's loop is repeated (as written in Algorithm 1). Its value is found by using the differential evolution algorithm [PSL06], a population-based meta–heuristic search algorithm (see Appendix A.2.1 for details).

### 3.2.4 Hyperparameters

We are now only left with the task of picking the best hyperparameters $\sigma^2, \sigma_N^2, \ell$ for the Matérn kernel. This is typically done by considering the marginal likelihood [RW05] (and Appendix A.1)

$$p(\boldsymbol{y}|\Theta) = \int p(\boldsymbol{y}|\boldsymbol{f}, \Theta) p(\boldsymbol{f}|\Theta) d\boldsymbol{f}, \tag{3.13}$$

where the prior $p(\boldsymbol{f}|\Theta)$ and the likelihood $p(\boldsymbol{y}|\boldsymbol{f}, \Theta)$ are Gaussian and the marginalization is done over the function values $\boldsymbol{f}$. Given the Gaussian nature of the likelihood and the prior, a closed form of the log marginal likelihood can be obtained (for the standard derivation of this formula see for example [RW05]):

$$\begin{aligned} \log p(\boldsymbol{y}|\Theta) = &-\frac{1}{2}\boldsymbol{y}^T \cdot (\boldsymbol{K} + \sigma_N^2 \mathbb{I})^{-1} \cdot \boldsymbol{y} \\ &-\frac{1}{2}\log\det\left(\boldsymbol{K} + \sigma_N^2 \mathbb{I}\right) \\ &-\frac{N}{2}\log 2\pi. \end{aligned} \tag{3.14}$$

where $N$ is the number of observations in the design matrix $\Theta$. In Eq. (3.14), the first term specifies how well the process fits the data, the second term instead acts as a regularization factor on the elements of the kernel matrix. When fitting the Gaussian process to a new set of points, the best hyperparameters $(\tilde{\sigma}^2, \tilde{\sigma}_N^2, \tilde{\ell})$ can be found by maximizing the log marginal likelihood in Eq. (3.14). For the optimization of $\log p(\boldsymbol{y}|\Theta)$, we use the quasi-Newton method `L-BFGS` [LN89] with multiple restarting points which proved to be efficient on the flat landscape of the likelihood (see Appendix A.1.1 for details).

## 3.3 Simulations Results

In this section we apply the Bayesian optimization to the QAOA parameters. We consider two well-known combinatorial problems defined on graphs, the Max-Cut and the Max Independent Set.

### 3.3.1 QAOA at low vs. large depth

We start by looking at the QAOA at depth $p = 1$ for the 6 nodes graph. It corresponds to a shallow circuit that depends only on two parameters $\boldsymbol{\theta}_1 = (\gamma_1, \beta_1)$. We consider the MIS problem, and we plot the landscapes of both the energy $E(\boldsymbol{\theta}_1)$ and the fidelity $F(\boldsymbol{\theta}_1)$ (Fig. 3.2(a) and (b), respectively) for values of $\gamma_1, \beta_1 \in [0, \pi]$ due to the symmetry of the problem. We see that the landscape of the energy, which is the function to minimize, is rather flat with two global maxima and minima, corresponding to the best solutions possible at $p = 1$.

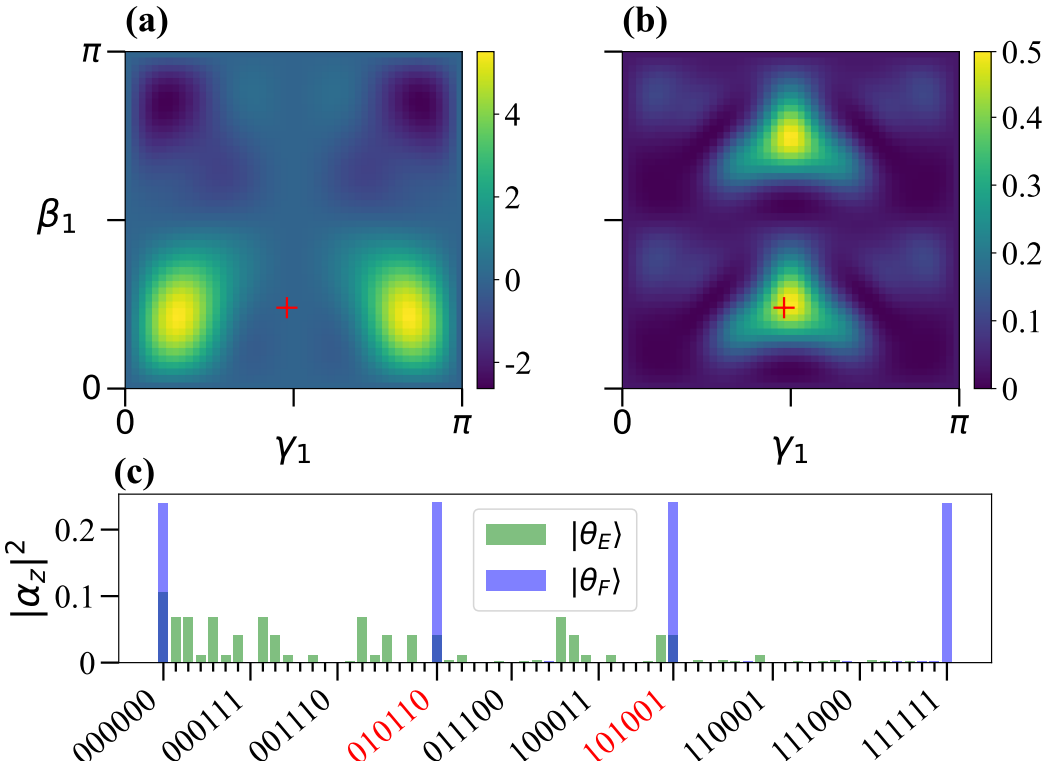



**Figure 3.2: QAOA at $p = 1$.** (a) Landscape of the energy $E(\theta_1)$ obtained on the 6 nodes graph solving the MIS problem. The red cross indicates the angles corresponding to the final state $|\theta_F\rangle$ with largest fidelity. (b) Landscape of the fidelity $F(\theta_1)$. (c) Squared amplitudes of the two states $|\theta_E\rangle$, $|\theta_F\rangle$. The solution bitstrings are highlighted in red. Values of the qubits are given in the order shown in the inset of Fig. 3.1(a).

Interestingly, we find that the QAOA state $|\theta_E\rangle = \sum_z \alpha_{z,E} |z\rangle$, corresponding to the parameters which minimize the energy, is not the state $|\theta_F\rangle = \sum_z \alpha_{z,F} |z\rangle$ with largest fidelity. To see how they differ we plot the squared amplitudes $|\alpha_{z,E}|^2$ and $|\alpha_{z,F}|^2$ of both states in Fig. 3.2(c) as histograms. The fidelity of $|\theta_F\rangle$ w.r.t. the solution $|z^*\rangle$ is, as expected, much larger than that one of $|\theta_E\rangle$, yet the latter has a lower energy because it has many non zero amplitudes along excited states with low energy. This unravels the problem of optimizing the QAOA parameters by only looking at the energy $E(\theta)$. There is a large concentration of excited states with energy comparable to the energy of the ground state, as shown in the histograms of panels (a) and (b) of Fig. 3.1. It is difficult to increase the amplitude corresponding to the solution when many other states can contribute with low values of the energy.

The difference between lowest energy and highest fidelity points is guaranteed to disappear theoretically for $p \to \infty$. For this reason, we apply Bayesian optimization to the problem and we show in Fig. 3.3 that the approximation ratio and fidelity both tend to 1 for $p \sim 12$. Yet, we already see a good performance at $p = 4$ where $R \sim 0.7$ and we have $F \sim 0.5$ meaning about a 50% chance of measuring the solution on the state obtained with QAOA.

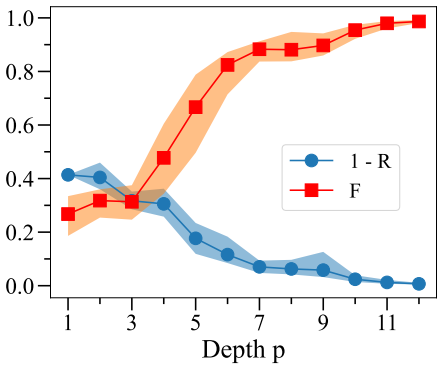



**Figure 3.3: Results increasing depth.** Average approximation ratio (plotted as $1-R$) and fidelity $F$ for increasing values of circuit depth from 1 to 12 over 50 runs. Shaded areas correspond to one standard deviation. Results were obtained on the 6 nodes graph of Fig. 3.1(a).

### 3.3.2 Comparing resources

Increasing the depth of a variational circuit increases the number of parameters that must be optimized. In turn, this is expected to increase the number of calls to the quantum circuit needed to reach a good approximate solution, which is a problem in the current NISQ era, since running a quantum circuit can be costly due to both state preparation routines and re-calibrations of the device.

For the noiseless circuit, it is very clear (see Figure 3.4) that the Bayesian approach can mitigate this problem better than any other tested techniques, from different points of view (such as number of calls, number of steps, number of measurements). Let us stress that this approach gives a fidelity that is always higher than the other methods at a fixed number of steps (and fixed $p$, see Figure 3.4c). All the simulations were run on python. The quantum circuit was simulated using the `qutip` package [JNN13], the Bayesian optimization part was built by expanding the class `GaussianProcessRegressor` of `scikit-learn` [PVG+11] and the other global optimizers were implemented using the standard `scipy.optimize` class [VGO+20].

### 3.3.3 Slow Measurements

The energy $E(\boldsymbol{\theta})$ is obtained by measuring the QAOA state after running the circuit: we refer to these two operations combined together as a "shot". By measuring on the $Z$ basis at each shot we get a bitstring, and we calculate its classical energy associated to the combinatorial problem. The precision in the reconstruction of $E(\boldsymbol{\theta})$ depends on the number of shots $N_S$. Since we consider this as a multinomial sampling problem we expect the variance of the reconstructed energy to depend on $N_S^{-1}$. In many scenarios of NISQ devices it is necessary to balance $N_S$ with the desired standard deviation. For this reason, we compare the average approximation ratio obtained with the exact energy (simulated) with the energy reconstructed with a limited number of shots.

We show in Fig. 3.5 such comparison with a number of shots $N_S$ equal to 1024, 128, 64, 16, 4. Looking at the approximation ratio $R$ (Fig. 3.5(a)) we see that taking

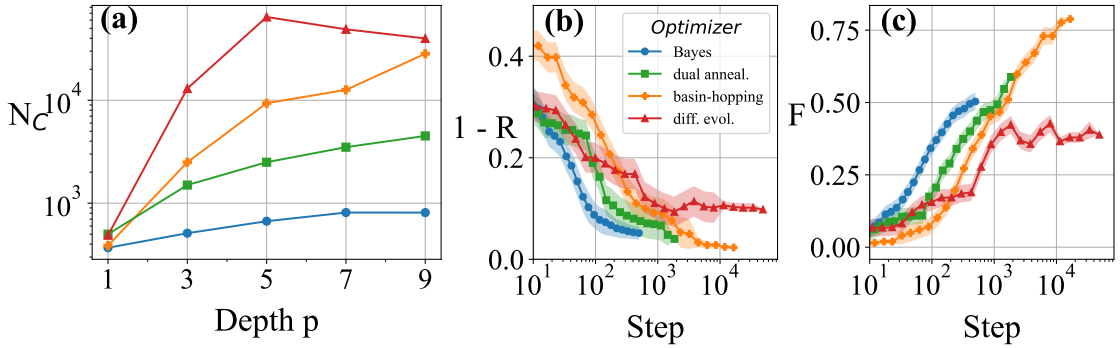



**Figure 3.4: Comparison among optimizers**. (a) The plot shows the average number of calls $N_C$ to the quantum circuit of each optimizer in order to obtain the same approximation ratio as the Bayesian optimization. (b) - (c) Average approximation ratio (plotted as $1-R$) and fidelity during the optimization with the different methods at $p = 7$ over 30 runs. Shaded areas correspond to one standard deviation. Results are obtained on the 10 nodes graph of Fig. 3.1(b).

$N_S = 128$ shots reduces $R$ by 5% w.r.t. $N_S = 1024$ and going to $N_S = 64$ reduces it by a further 5%. This behaviour then stops and even reverses its trend. In fact, we even see an average increase going from $N_S = 16$ to 4. This is understandable since the reconstruction of the energy with as little as 4 shots is not indicative of the real energy of the state. Specifically, from a final QAOA state we might sample the solution bitstring 2, 3 or even 4 times out of 4 and the expectation of the energy on these three samplings would be very different. This behaviour is indeed confirmed also by the fidelity in Fig. 3.5(a) which follows the same trend as the approximation ratio.

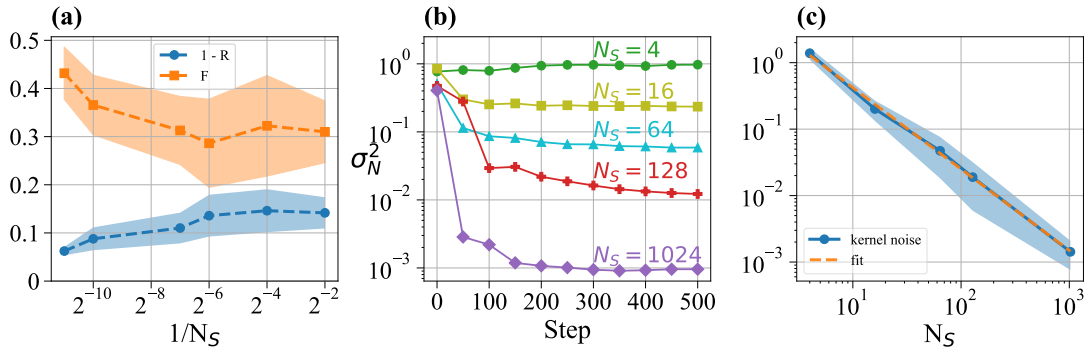



**Figure 3.5: Slow measurements.** (a) Average approximation ratio (plotted as $1-R$) and fidelity ($F$) as a function of the number of shots $N_S$. The $1/N_S = 0$ points indicates the exact evaluation of the energy $E(\boldsymbol{\theta})$. Shaded areas correspond to one standard deviation. (b) Kernel noise $\sigma_N^2$ learned by fitting the data at each step of the optimization for different number of shots $N_S$. (c) Average kernel noise learned by the Gaussian process as a function of $N_S$ (blue circles). The plot also shows a linear fit ($\sim 1/N^{1.1}$, orange line) of the logarithm of the data.

To have a better understanding of how the algorithm adapts to the sampling noise, we look at the kernel noise parameter $\sigma_N^2$ which is learned by the Gaussian process

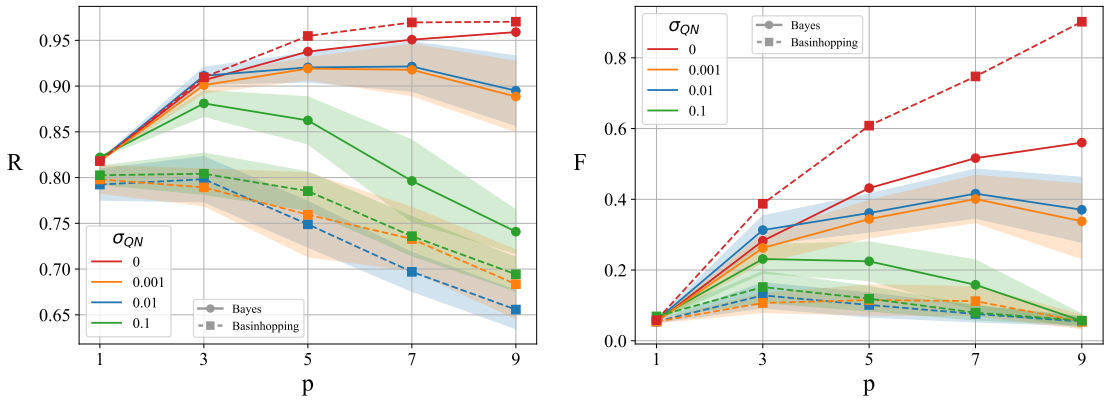



**Figure 3.6: Approximation ratio $R$ and fidelity $F$ for different values of the quantum noise $\sigma_{QN}$.** The noise is simulated by adding random Gaussian noise with mean zero and standard deviation $\sigma_{QN}$ to the variational parameters $(\boldsymbol{\gamma}, \boldsymbol{\beta})$. The plot shows the effects of the noise $\sigma_{QN}$ on the final obtained approximation ratio $R$ (fidelity $F$) as a function of the QAOA depth $p$ for different $\sigma_{QN}$. Shaded areas correspond to one standard deviation. The results are obtained on the graph with 10 nodes in Fig. 3.1(b).

during the fitting at each step of the optimization (see Appendix 3.2.4 for details on the noise hyperparameter). The plot in Fig. 3.5(b) shows that, after an initial phase, the kernel noise sets at a specific value at around 400 steps. In addition to that, the lower the number of shots the larger the noise parameter learned. In fact, by fitting the average kernel noise found at the end of the training (Fig. 3.5(c)) we obtain that $\sigma_N^2$ follows a power law with $N_S^{-1.1}$. This trend is comparable to the expected trend for the variance $N_S^{-1}$ of the reconstruction of the energy. This shows that the Gaussian process adapts to sampling noise.

### 3.3.4 Noise Simulation

Another relevant issue in the state-of-the-art NISQ devices are the sources of quantum noise which can interfere with the quantum circuit. Every device has different sources of noise depending on the underlying technology. In order to simulate it without specifying the device technology we add random noise on every QAOA parameter. In this way Eq. (3.2) for the Max-Cut problem is modified as:

$$|\boldsymbol{\theta}\rangle = \prod_{l=1}^{p} e^{-i\sum_i \beta_l^i X_i} \; e^{i\sum_{\langle i,j\rangle} \gamma_l^{(i,j)} Z_i Z_j} |+\rangle, \tag{3.15}$$

where $\beta_l^i$ and $\gamma_l^{(i,j)}$ act differently on every qubit/edge of the graph at every layer because they are affected by Gaussian random noise with mean zero and standard deviation $\sigma_{QN}$. The noise model we are considering, although very simplified, can be considered as an example of coherent control errors that can be caused e.g. by gate over-rotations due to gate-time mis-calibrations. In Fig. 3.6 we plot $R$ and $F$ as a function of depth at different values of $\sigma_{QN}$ on the 10 nodes graph for the Max-Cut problem.

By increasing $\sigma_{QN}$ and $p$, we expect to obtain a worse approximation ratio $R$ because the error accumulates along the circuit as the number of parameters grows. Indeed, as we can see in the figure, from $p \geq 5$ the obtained $R$ decreases w.r.t. the noiseless case, decreasing even by 20% for $p = 9$ with $\sigma_{QN} = 0.1$. Considering $\sigma_{QN} = 0.001, 0.01$, both $R$ and $F$ grow/remain stable up to $p = 7$ which indicates that, for shallow circuits, Bayesian optimization is robust against noise.

We care to stress that the case $\sigma_{QN} = 0.1$ was considered in order to show the effect of an exponential growth of machine noise. Realistically, a Gaussian white noise with variance 0.1 affecting each of the parameters (in range $[0, \pi]$, see Appendix A.1.1), would completely destroy the state preparation. In fact at $p = 9$ the fidelity $F$ is the same as $p = 1$ (Fig. 3.6).

We compare the results of our algorithm with the second best performing algorithm of Fig. 3.4, basin-hopping. It is clear that when subjected to noise this algorithm performs poorly: considering the approximation ratio, basin-hopping shows barely any improvement with respect to the depth of the circuit with the results starting to plummet from $p = 3$ (Fig. 3.6). Most importantly, the fidelity peaks at $p = 3$ with $F \simeq 0.15$ (Fig. 3.6) and then remains contained under this value. About the seemingly increase in fidelity with the noise that can be seen at $p = 3$, we also notice that the behaviour inverts going up to $p = 7$ so we do not consider it relevant and assume that this means that the results are too random and basin-hopping is thus non-reliable. The poor performance in terms of fidelity confirms that basin-hopping, while being an effective algorithm in the noise-free scenario – visible thanks to the high, yet costly, performance at $\sigma_{QN} = 0$ (cf. Fig. 3.4) – is not apt for optimization in the presence of noise. This is probably due to the fact that basin-hopping is a global optimizer that exploits a local gradient-based optimization routine (see Appendix A.2). Calculating gradient in the presence of noise is in fact non-optimal since even a small variation of the parameters can impact greatly the evaluation of the function, returning a gradient that does not represent the local structure of the landscape.

# 4 | QAOA with Bayesian Optimization on Rydberg Atoms

In this chapter we introduce quantum computation with neutral atoms, one of the many technologies that is currently being investigated to implement a quantum computer or, better, a Quantum Processing Unit (QPU). The reason for this multitude of methods is that each technology presents pros and cons and at the moment they all seem comparable to one another. In this chapter we present this type of platforms, focusing on how they can be used for an analog version of QAOA.

The core part is presenting the data we collected from running the Bayesian Optimization on the Rydberg QPU of the company PASQAL which was one of the main goals of my PhD. We start by explaining how to run QAOA in the Analog computation context that is allowed at PASQAL highlighting the main differences with the digital approach to QAOA presented in the first section.

## 4.1 Quantum computing on a Neutral atoms device

The main advantages brought in with this method, to cite a few, are: the possibility to manipulate the positions of the atoms with ease, mediated with tweezers both in 2D and 3D [BLdL+18], and ability to create entanglement between qubits.

In this promising technology, an array of atoms is trapped with tweezers and the electron living in the most excited state is allowed to transition between two states. This two states system is considered as a qubit. The main properties of this qubit is that it does not interact with its nucleus and it can be coupled to a state with large excitation that prevent two qubits close to each other to be excited at the same time. This is the so called *Rydberg blockade* effect and it can be exploited to create entanglement between qubits. By applying a set of laser pulses to the qubits, we can show how to perform the basic gates of quantum computation and thus perform standard "digital" quantum computation. In a similar way we can also perform "analog" quantum computation by applying pulses to all the qubits at the same time. In particular, these architecture gives the possibility to manipulate the positions of the atoms with ease, mediated with tweezers both in 2D and 3D [BLdL+18] and scaling possibility in the order of the thousands [SWM10]. These properties have attracted much interest by both academia and companies [HBS+20]. It was, in fact, shown that Rydberg atoms are able to simulate quantum phases of hundreds of sites [SLK+21], [CBB+23], up to 256 [EWL+21], and to perform algorithms as

quantum optimization [ADL+23, EKC+22].

### Rubidium atoms

The core component of quantum computation is the qubit, in the Rydberg atoms platforms of PASQAL, a qubit is made with a Rubidium atom. Rubidium's atomic number is 37 with electronic configuration $5s^1$, so its energy levels are completely filled and has one single electron in the 5S shell. The isospin is $I = 3/2$ and the angular momentum is $J = \pm 1/2$. This gives an hyperfine structure of $F = J + I = 1, 2$ which identifies the two splitting states of the groundstate. These two excited states of the atom are taken as reference for the $|0\rangle$ and $|1\rangle$ state. The atoms/qubits are both positioned and controlled with light. The valence electron is also coupled to a highly excited state $|r\rangle$, usually with energy number 60. This is the Rydberg state that allows to create entanglement between qubits as shown in the following section.

### Rydberg blockade

A Rydberg atom is an atom where one or more electrons are pushed to a very high energetic quantum number $n$. In the case of rubidium we go from $5s$ state to $70s$ state. In this regime there can be a strong Van der Waals coupling between two excited electrons. So, imagine a situation where an electron in a groundstate $|g\rangle$ like the rubidium $5s$ is coupled to an external field with a Rabi Frequency $\Omega$ which makes it jump to the Rydberg state $|r\rangle$. If we have two electrons at distance $R$ and we excite the groundstate $|gg\rangle$ to $|rr\rangle$ we will have a Van der Waals term $C_6/R^6$ (with $C_6 \approx 4000$ in rubidium at $5s$) which makes the energy of $|rr\rangle$ larger than the other combinations like $|gr\rangle$ (which are not effected by the distance $R$). This makes it impossible to reach the state $|rr\rangle$ with the Rabi frequency and thus it is not accessible. Consequently, the two electrons system oscillates between $|gg\rangle$ and an entangled pair $|gr\rangle + |rg\rangle$ with enhanced Rabi frequency $\sqrt{2}\Omega$. This happens in the regime where the Van der Waals interaction term is larger than the energy of the Rabi frequency, that is $\hbar\Omega << C_6/R^6$ which happens when the distance between electrons is $R << R_b$ with $R_b = (C_6/\hbar\Omega)^{1/6}$.

The main difference from other QC platforms [HBS+20], like superconducting, is that the register of qubits in neutral atom systems is created at the beginning of the computation and destroyed at the end. For this reason we can follow three main steps of computation:

- Register preparation
- Quantum processing: Digital and Analog
- Readout measurement

The three operations combined require about 200 ms giving a not so large repetition rate in the order of $1 \sim 5$Hz, the quantum processing only 100 microseconds or less.

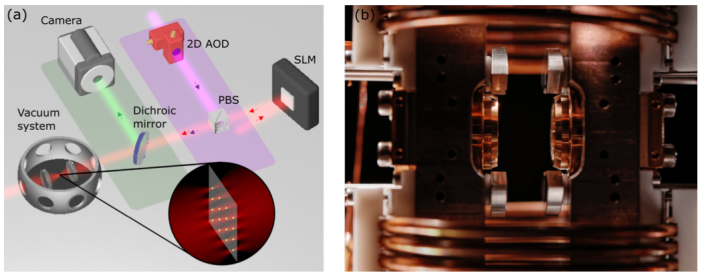



**Figure 4.1: Setup of the quantum computer**. Three light sources control the qubits: red light traps the qubits, purple light are the optical tweezers that rearrange their position, green light is the fluorescence camera. Credit for the picture: [HBS+20]

### 4.1.1 Register loading and Readout

We start with a dilute vapor of Rubidium inside an ultra-high vacuum system at room temperature. Then a cold ensemble of $10^6$ atoms in 1 $mm^3$ volume is prepared. After that, a trapping laser system isolates every atom at a distance of about one micrometer (red light in Fig. 4.1), each atom is controlled by an optical tweezer.

The camera (green light) tells the luminescence of each atom to see where they are. The atoms are moved by the tweezers from site to site with a 99% success rate. An algorithm decides what moves to make to design a desired shape on the fly. The tweezers action takes about 1 ms.

**Register Readout** Reading the state of qubits is done with a camera that gives green fluorescence to $|0\rangle$ state qubits while $|1\rangle$ state qubits result dark. The efficiency of this operation is reported to be of 98.6% [HBS+20] in the white paper by PASQAL, comparable to superconducting quantum computers.

### 4.1.2 Digital vs Analog Quantum processing

State-of-the-art platforms are currently running algorithms in two modes: digital and analog. The main difference is that the former entails acting on single qubits or couple of qubits at the same time while the latter means acting on all the qubits at the same time. At the moment of writing and running the experiment on the PASQAL machine only the analog approach was implemented so we will focus mostly on the latter. A third option, measurement-based, is the main topic of the remain of the chapter.

For a quantum computer to be universal it needs to correctly implement single qubit arbitrary rotations and an entangling gate like CNOT because they virtually realize any possible quantum circuit.

#### Single qubit rotation

The beams interact with the atoms and are controlled by three parameters:

1. Rabi Frequency $\Omega$: is the strength of the laser, it is proportional to the amplitude of the laser electric field and the electric dipole of the qubit;

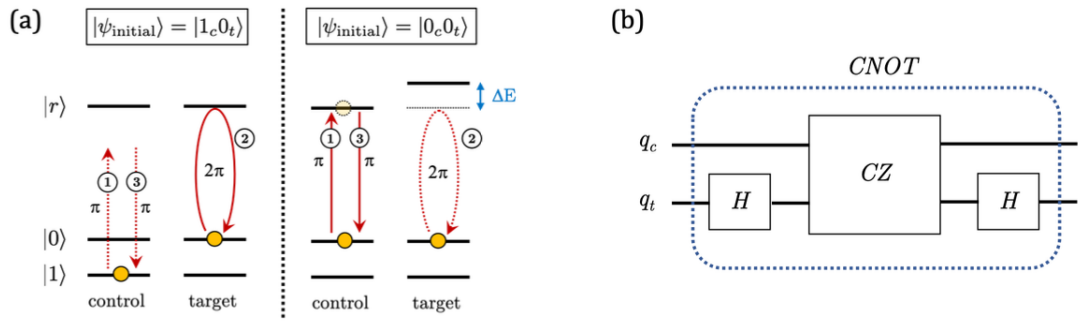



**Figure 4.2:** Implementation of the CNOT with 3 pulses. (a) the sequence of pulses that creates entanglement. Pulse 1 and 3 act on the control qubit applying a $\pi$ rotation while pulse 2 acts on the target qubit with a full $2\pi$ rotation. (b) Inserting a $CZ$ between consecutive Hadamard gates produces a $CNOT$ gate. Credit for the picture: [HBS+20]

2. Detuning $\delta$: the difference between the qubit resonance and the field frequencies;
3. Relative phase $\phi$.

Applying this laser to a qubit produces an overall Hamiltonian acting on qubit $q_i$:

$$H_{q_i} = \hbar\Omega(t)\left[\cos(\phi)\sigma^x_{q_i} - \sin(\phi)\sigma^y_{q_i}\right] - \hbar\delta(t)\sigma^z_{q_i} = \hbar\mathbf{\Omega}(t)\cdot\boldsymbol{\sigma_i} \tag{4.1}$$

where $\mathbf{\Omega}(t) = (\Omega(t)\cos(\phi), \Omega(t)\sin(\phi), \delta(t))$. This means that if we put the phase equal to zero $\phi = 0$ and make the pulse time-independent $\Omega(t) = \Omega$, we have

$$H_{q_i} = \hbar\Omega\sigma_{ix} - \hbar\delta\sigma_{iz} \tag{4.2}$$

which corresponds to a rotation around the vector $(\Omega, 0, -\delta)$, on the $X - Z$ plane of the Bloch Sphere. We can also generate other one-qubit gates acting with a constant $\Omega$ pulse for a time $\tau$:

- NOT GATE: Simply put $\delta$=0 and $\Omega\tau = \pi$;
- HADAMARD: Setting $\phi = 0$, $\delta = \Omega$ and $\Omega\tau = \pi/2$.

### CNOT

CNOT is implemented on Rydberg atoms by creating first a $CZ$ gate and adding to Hadamard gates. Consider two qubits at a distance lower than the Rydberg radius. Both are coupled to a pulse with an $\Omega$ Rabi frequency that puts a qubit in $|0\rangle$ in oscillation with $|r\rangle$. It was shown [HBS+20] that we need only 3 pulses to create the $CZ$ gate: a $\pi$ rotation on the control qubit (the first one), a $2\pi$ rotation of the target and then another $\pi$ on the control. See Fig 4.2 for more details.

### 4.1.3 Analog Quantum Processing

In the analog framework we couple the qubits only to the groundstate-Rydberg transitions, therefore we only address two states: the groundstate, the $|0\rangle$ state, and the Rydberg state, that will be from now on called $|1\rangle$. All the qubits receive the same pulse that couple the $|0\rangle$ and the Rydberg state. This results in an evolution of the whole register under the effective Ising Hamiltonian:

$$H(t) = \frac{\hbar}{2}\Omega(t)\sum_j \sigma_j^x - \hbar\delta(t)\sum_j n_j + \sum_{i\neq j}\frac{C_6}{r_{ij}^6} n_i n_j \tag{4.3}$$

where $n_j = (1 + \sigma_j^z)/2$ is the state occupancy of the Rydberg state. The first two terms are induced by the laser and can be adjusted by changing its intensity and frequency, for spins they correspond to the interaction with, respectively, the perpendicular and parallel component of an external magnetic field which makes the spin turn according to the Larmor precession. The third term is once again due to the Van der Waals interaction that prevents neighbouring atoms to both have an electron in the $|r\rangle = |1\rangle$ state. Clearly, it takes the same form of the usual interaction between spins of the Ising model.

In analog mode there is no freedom in accessing single qubits, thus it all boils down to deciding the frequency $\Omega, \delta, \phi$ of the laser and its duration. We will give more details on this in Sec. 4.2.

### 4.1.4 Noise Sources

Quantum computers like the Rydberg atoms QPU are isolated to perform computations. Therefore, performing quantum computation means to deal with many sources of noise that greatly affect efficiency. We will present here the most significant sources that can also be simulated on the open-source package developed by PASQAL: Pulser [SGD+22].

#### Doppler damping

Since the atoms are cooled to a temperature in the order of $T \sim 50\mu$K they experience a Doppler-shifted detuning frequency. At the simulation level this is obtained by adding a random noise to the detuning $\delta$ acting on each qubit. The noise is generated from a normal distribution centered at zero with deviation $\sigma_D$. Then the detuning of each qubit is modified by adding this qubit-specific noise. The $\sigma_D$ standard deviation is calculated as:

$$\sigma_D = K_{EFF}\sqrt{\frac{K_B T}{M}} \tag{4.4}$$

with $KEFF = 8.7\mu m^{-1}$, $k_B$ is the Boltzmann constant, $M = 1.45 \times 10^{-25}$ is the mass of Rubidium$^{87}$ and $T$ is the temperature. The default temperature is T = 50$\mu$K which gives $\sigma_D = 0.60014$.

### Amplitude

This dumps down the value of the amplitude ($\Omega$) on each qubit depending on its distance from the center of the graph, which is where the laser is pointing for a global pulse. The position dependent modification to $\Omega$ is given by:

$$\Omega(r) = \Omega \times \varepsilon \times e^{-(\frac{r}{w_0})^2} \tag{4.5}$$

where $\varepsilon \sim N(1, 10^{-3})$, $w_0$ is the laser waist and $r$ is the distance. We can calculate that this amplitude error decreases $\Omega(r)$ by 5% for qubits placed at least 33$\mu$m away from the origin. This will not be a problem for the graphs considered in this work that at the most distance about 20$\mu$m from the center of the beam.

### SPAM

With SPAM we mean "State preparation and Measurement" errors, the ones that we tackled the most in this work. For the state preparation part there is always a probability $\eta$ of not loading correctly the initial state of each qubit. At the simulation level, for every qubit a random number is extracted from 0 to 1, if it is less than $\eta$ then that qubit becomes a bad atom. A bad atom is an atom which is simply excluded from the simulation, so it does not receive a sequence and it is not measured.

For the measurement part we have two sources of error:

- False positives: an atom in $|0\rangle$ is mistakenly seen as in $|1\rangle$. We associate the $\epsilon$ parameter to this error.
- False negatives: an atom in $|1\rangle$ is mistakenly seen as in $|0\rangle$. We associate the $\epsilon'$ parameter to this error.

common values obtain through calibration of the PASQAL's QPU even at the moment of the experiments are $\epsilon \sim 5\%$ and $\epsilon' \sim 8 - 10\%$.

### Dephasing and Depolarizing channels

There is a final component of effective noise channels not due to the lasers or the preparation scheme but to the interaction with the environment. The simulation of this noise sources requires to use the density matrix representation of the state and to solve the noisy Lindblad equation instead of the Schrodinger's one. The two main sources of this noise are:

- Dephasing channel: it represents the loss of coherence (the non-diagonal terms in the density matrix) and is implemented with random $Z$ rotations happening with probability $p$.
- Depolarizing channel: a more generic error representing the interaction with the environment that erases the quantumness of the state. At the simulation level is represented by an evolution that evolve the density matrix to the completely mixed state with probability $p$.

### 4.1.5 Pros and Cons

The use of Rydberg blockade introduces two advantages then:

1. Large connectivity, which brings lower overhead in the number of qubits with respect to general nearest neighbour structures such as superconducting qubits;
2. Easy implementation of 2-qubits and more-than-2 qubits gates like Toffoli (which requires only 7 pulses);
3. Less computation time (1$MHz$ clock for the computation);

Other advantages already mentioned are:

1. Possibility of arranging the qubits in 2D or 3D shapes;
2. High scalability in the number of qubits, limited only by the number of available optical tweezers.

The negative side of this platform are mainly:

1. Low repetition rate, that is currently in the order of the $Hz$ making it very costly, in terms of time, to make many acquisitions of the final state.
2. Measurement error which influences greatly the reconstruction of the wavefunction with many bitstrings wrongly measured.
3. Destructive measurement, at the moment of writing the only measurement allowed consists in getting rid of all the qubits in Rydberg and measuring the remaining ones on the ground-state. This completely destroys the state of the system.

## 4.2 Analog QAOA on Rydberg atoms

The purpose of this section is to show the optimization loop that was implemented on the real machine and understand how QAOA needs to be modified when run in analog mode. This means applying the same pulse to every qubit and thus not being able to compile precisely the Hamiltonian that we want. Nevertheless, there is a way around that: considering problems that are directly encoded in the atoms' evolution.

The experimental loop of QAOA is represented in Figure 4.3. The main functioning runs exactly in the same way: a set of parameters is fed to the quantum part of the algorithm and a value proportional to the energy of the system is given back compared to the simulations performed in Chapter 3. In addition to that, we have to consider a few intermediate steps, in particular the problem, the graph loading, the sequence of pulses, the measurements of the state and the post processing of the statistics.

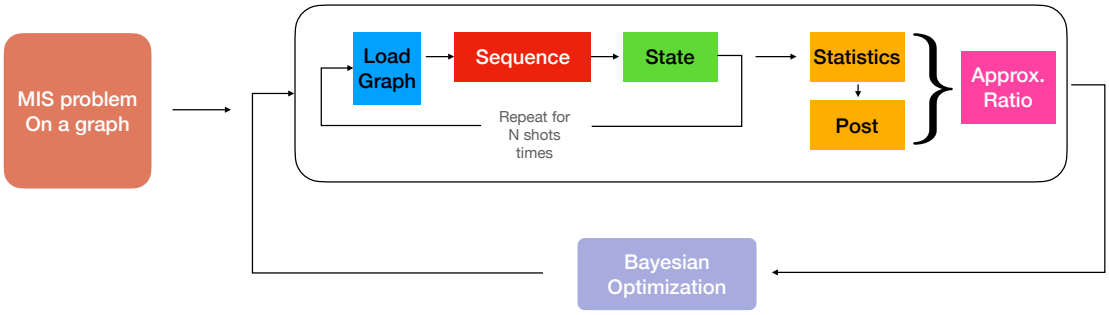



**Figure 4.3: QAOA loop**. We start with the MIS problem defined on a chosen graph. The optimization loop consists in the quantum part and the classical part. In the quantum part, we start loading qubits on the register in positions that encode the graph of our problem, apply a sequence of pulses with parametric duration, acquire the state through measurements and repeat this for a number of shots. From all the measurement we construct a statistic of the final state, we apply a post-procedure if necessary and extract the energy. The parameters and energy are then fed to Bayesian Optimization.

### 4.2.1 The MIS problem

We decided to tackle the MIS problem, introduced in Chapter 3. There are multiple reasons for this. The first one is that the Rydberg blockade naturally prevents neighbouring qubits to be excited at the same time and this is also a condition of the MIS problem. This cuts down also a portion of the Hilbert space that would not hold the solution state.

The second one is that QAOA needs the alternation of the problem Hamiltonian and the mixing Hamiltonian, since we are in analog mode we cannot pick a driving Hamiltonian that suits us. Luckily, MIS is an ideal problem because the Raman pulse induces an evolution by a Hamiltonian such as:

$$H_\delta = \delta \sum_i \sigma_z^i + U \sum_{(i,j)\in E} n_i n_j, \tag{4.6}$$

with $n_i = (1+\sigma_i)/2$, which is the quantum translation of the MIS problem classical Hamiltonian, i.e.:

$$H_{MIS} = \sum_i x_i + U \sum_{(i,j)\in E} x_i x_j$$

.

### 4.2.2 Graph Loading

As we have seen in Sec. 4.1.1 to load qubits into the platform means to isolate each qubit in a optical tweezer and as we know this process has a success rate of 50% meaning that the optical tweezer is either empty or filled with one atom. After the loading the atoms are generally rearranged to the desired positions. In our project, we need to reproduce a graph in the atoms' platform. That means that we need each node in the graph to be represented by an atom and all the neighbouring nodes to have the same distance, in order to experience the same Rydberg blockade.

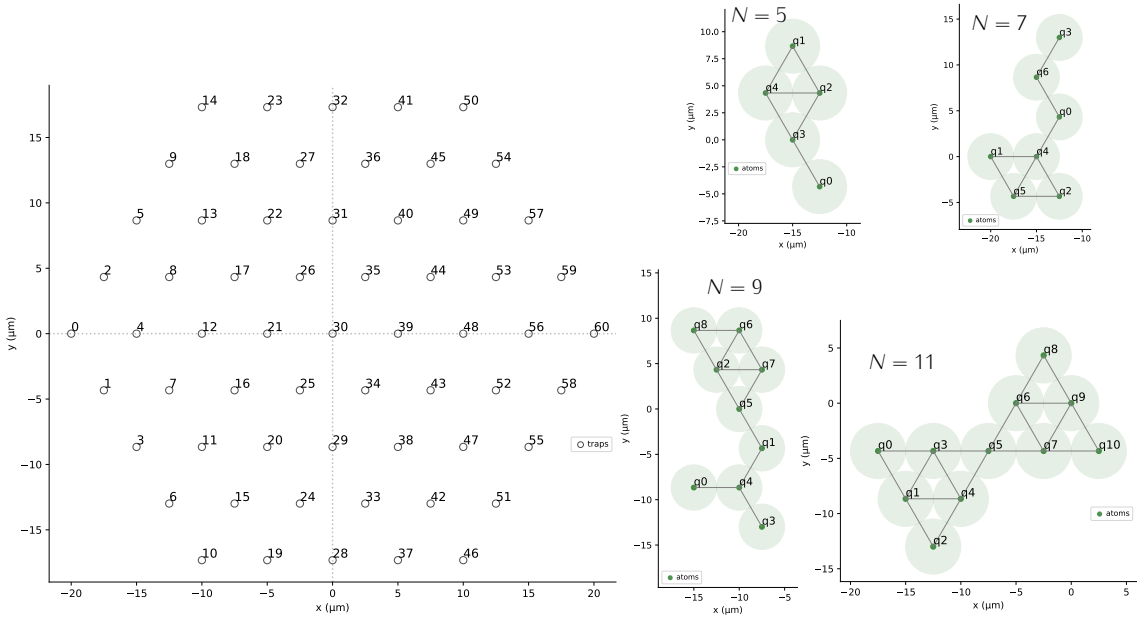



**Figure 4.4: Layout and graphs** Triangular layout (left) with 60 traps. (right) Some of the graphs used for simulation and for RUN1 and RUN2 (see Sec. 4.4) loaded on the triangular layout of traps on the left.

A convenient way to do this is with a trap layout that is already set to a specific geometry. For our experiment we decided to use a triangular lattice trap layout, which is shown in the upper corner of Figure 4.4. To create a graph, we only need to decide the traps of the triangular layout.

For the simulation and the experiment we used graphs with an increasing number of qubits from 4 to 10, the $N = 6$ was used both in simulation and in the experiment while $N = 11$ was used only in the experiment because the full simulation of noise is very memory-demanding. The graphs are shown in Figure 4.4.

### 4.2.3 The sequences

The pulses available for analog quantum computation are the Raman and the Rydberg pulses and were introduced in chapter 4. Both pulses have global addressing meaning they address all the qubits at the same time. The difference is that the Raman pulse applies a phase to each qubit, whereas the Rydberg pulse also introduces a $\sigma_x$ rotation on the qubits. The pulses can be shaped with varying $\Omega(t)$, $\delta(t)$ but we decided to keep them constant in time for simplicity.

As it is customary in QAOA we decided to add an initial driving Rydberg pulse with a total angle of $\pi$. With a selected constant $\Omega = 2\pi$ MHz this means that the initial pulse lasts for 250 ns. This is similar (but not equivalent due to the interaction term) to initializing all the qubits to $|+\rangle$. Following that, we alternate the Raman and the Rydberg pulse $p$ times where $p = [0, 5]$ for parametric times: $\theta_1^p$, $\theta_2^p$. Thus, we have the customary $2p$ parameters of QAOA, the Rydberg pulse being always the last one before measurement because adding a phase would not affect it. An example sequence showing a QAOA instance of depth 1 is shown in Fig. 4.5

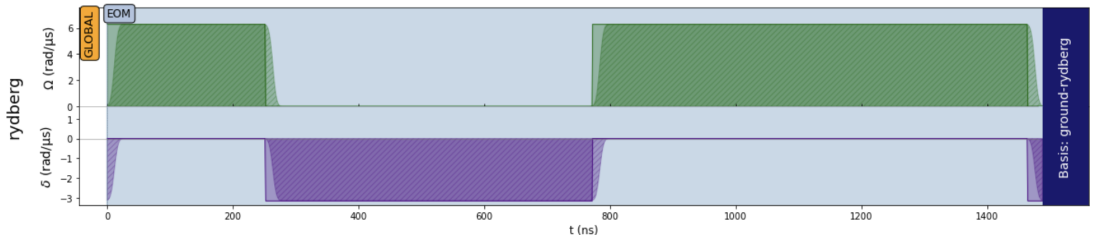



**Figure 4.5: Example sequence**. Sequence sent to the Rydberg global channel addressing all the qubits. The Ω pulse (green) is coupled to and drives the transition from the ground state $|0\rangle$ to the Rydberg state $|r\rangle$. The detuning pulse (purple) applies a phase to the qubits. On the $x$ axis we have the time. A typical instance of QAOA corresponds to a set of parameters deciding the length of this two alternated pulses repeated for $p$ times.

There are other specifics of the PASQAL's architecture that were taken into account for the optimization of the sequences:

- The minimum pulse is 100 ns. This is because even for a constant pulse there is a ramp time during which the signal frequency increases from 0 to the desired value;
- The maximum length of a pulse was set to 1000ns. This is because with $\Omega = 2\pi$ MHz a pulse of 1 $\mu$s corresponds to a $2\pi$ angle.
- The total sequence of pulses cannot exceed 4000 ns due to decoherence (3750 considering the initial pulse). This does not affect the choice of parameters at low depth ($p = 1, 2$) but for $p \geq 3$ we had to constrain minimization with the inequality $\sum_p \theta <= 3750$.

All in all, we decided to restrict the depth to a maximum of $p = 5$, that is 10 parameters and the parameter search was limited by the linear constraint just mentioned.

### 4.2.4 State reconstruction

The final state of a circuit is obtained with a fluorescence image that shows green spots for each qubit int the $|0\rangle$ state while the qubits on $|1\rangle$ (that is, the Rydberg state $|r\rangle$ are lost and therefore) will be dark. As already explained above there is an important source of error at readout that can affect greatly the signal. At the time of launching the experiments there was a $\varepsilon \sim 5\%$ of false positive (mistaking a 0 for a 1) and almost $\varepsilon' \sim 10\%$ of false negative (mistake 1 for a 0). In the next section we will see how simulations and the real experiment were affected by this noise and how we dealt with it. This will introduce a simple post-processing step inside the optimization loop shown in Fig. 4.3.

Another element restricting our research was the number of shots available in total. When working on a time budget it is important to select a priori how many shots to use to reconstruct the energy of the state. Our analysis of Bayesian Optimization in Chapter 3 showed us that we can go very low with the number of circuit repetition. This is a good sign since we have a limited number of shots and the repetition rate

of PASQAL platform is in the order of $1 \sim 3$ Hz. We only need to check if this holds also in the Rydberg atom's case: for this reason we ran simulations and ran the optimization with as few shots as 16 per step.

## 4.3 Simulation Results

We end the chapter with a set of results that combine the Bayesian Optimization with the Rydberg platform introduced in this chapter. We present two sets of results: simulations and experiment. The simulations were run using `Pulser` [SGD+22]: the open-source python package of PASQAL developed for simulating sequences of pulses on registers of neutral atoms and especially to simulate the specifics platforms that the company offers. The experiment, instead, shows the results we obtained running a set of sequences and optimization loop on the real platforms in collaboration with the team of PASQAL.

The simulations were ran to gain intuition on how the optimization works on the Rydberg's QPU. During optimization we keep track of many parameters related both to the quantum state and the Bayesian Optimization. In this first part of the results we only concentrate on the state and plot what we call the QAOA parameters. We defined them as:

- Approximation Ratio $R$, We define it as

  $$R = E(\boldsymbol{\theta})/E_0$$

  , where $E(\boldsymbol{\theta})$: is the energy obtained running QAOA with parameters $\boldsymbol{\theta}$ and $E_0$ is the energy of the solution. We often use also the inverse approximation ratio $1 - R$;

- Fidelity $F$: calculated as

  $$F = \langle \psi(\boldsymbol{\theta}) \, | S_0 \rangle$$

  where $|S_0\rangle$ is an eigenstate in the computational basis representing the solutions bitstring (or a superposition of multiple ones if the solution is not unique)

- Solution Ratio $Sr$: it is defined as

  $$S_r = \frac{p(S_0)}{p(S_{2^{nd}})}$$

  where $p(S_0)$ is the probability of measuring the solution state $|S_0\rangle$, while $p(S_{2^{nd}})$ the probability of the second most likely state. Therefore $S_r$ is a measure of how much the solution stands out between the other states. $S_r \in [1, \text{inf}[$ if $S_0$ is the most measured state, we set it to 0 otherwise by choice.

First of all, we are interested in seeing how the algorithm scales with the system size. For this reason we averaged the QAOA parameters for 10 runs with number of qubits varying in $N \in [4, 10]$ – some of the graphs are shown in Fig. 4.4 – and plotted the results in 4.6.

By comparing the approximation ratio $R$, the fidelity $F$ and the solution ratio $S_r$ we notice a diminishing performance with the increase of the number of qubits. This

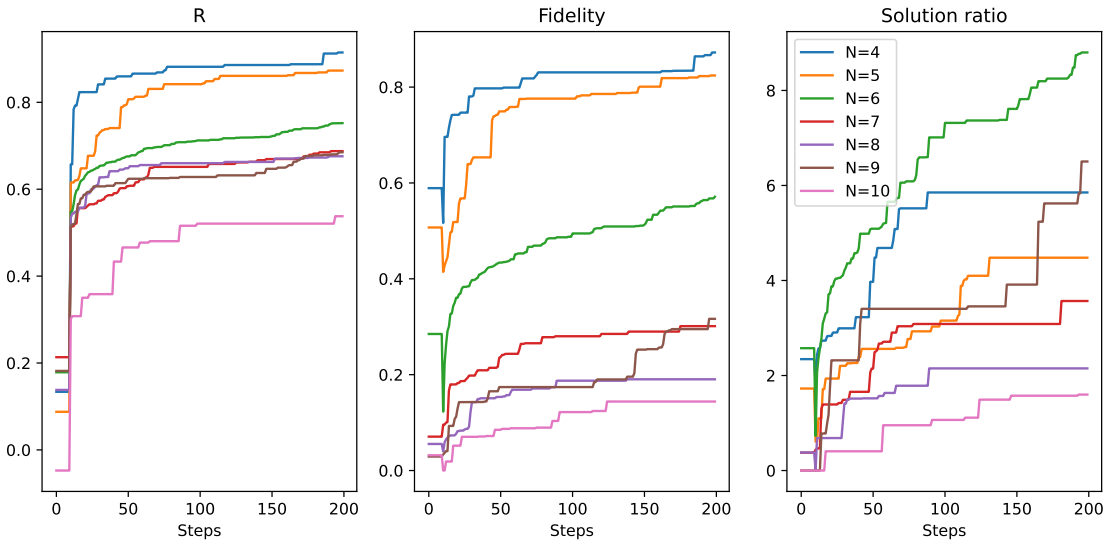



**Figure 4.6: QAOA simulated on *Pulser* at different graph sizes.** Averaged $R$:approximation ratio, $F$:fidelity and $S_r$:solution ratio (explained in the text) over 10 runs at different graph sizes $N \in [4, 10]$.

is not surprising since a bigger graph means a more difficult optimization and the presence of noise increases its effect when enlarging the graph. The simulations were ran considering 10 initial training points and 190 steps of optimization with 1028 shots. The parameters $R$ and $F$ do not increase much after the $90^{th}$ step contrary to $S_r$ that for the specific cases of $N = 5, 6, 9$ even doubles in the second half of the optimization.

We compare QAOA results also for different number of shots shown in Figure 4.7. From this results we see that it is possible (if not even better) to optimize with a lower number of shots. Yet, the fact that the 1028 shots optimization (which we assume to be an almost loyal reconstruction of the real state) has always a worse performance than the others means that the state reconstruction with few-shots has a tendency to over-estimate the energy.

## 4.4 Experiment Results

We now turn to the results obtained directly to the real QPU of PASQAL. We were allocated 18 total hours of computation, the QPU at PASQAL at the moment of writing run at a rate that oscillates from $1 \sim 5$ Hz, the results we gathered were ran with an effective rate of $0.86 \sim 1.15$ Hz. In total we were allowed to use approximately $\sim 70000$ shots. We decided to split them into three parts:

- RUN 1: 2 hours (6168 shots) to run specific sequences of pulses to check if the results are aligned with the simulations
- RUN 2: 6 hours (14080 shots) to run a few closed-loops of optimization
- RUN3: 12 hours (50000 shots) to run the optimization on a large graph

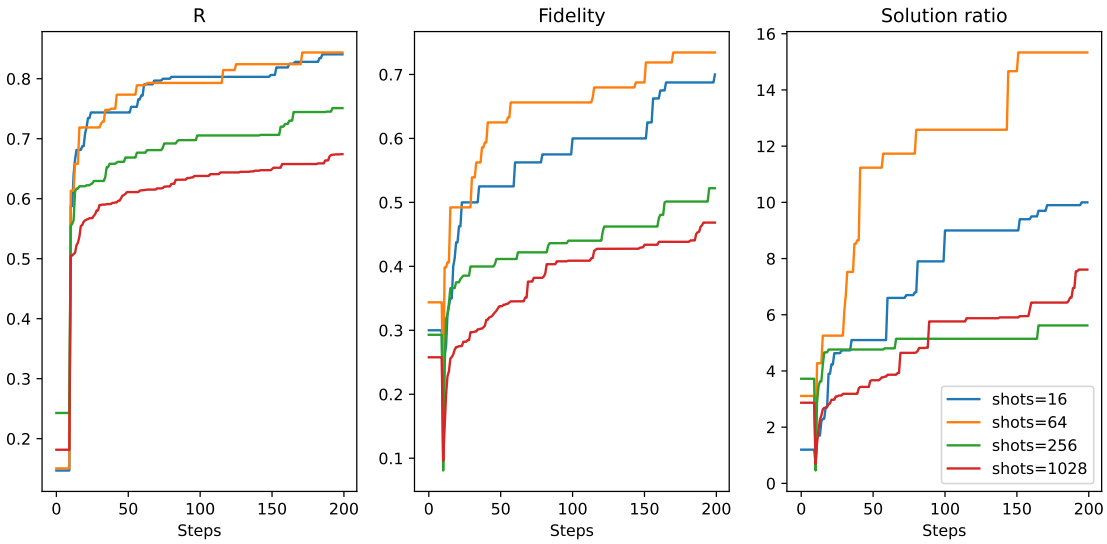



**Figure 4.7: QAOA simulated on *Pulser* at different number of shots**. Averaged $R$:approximation ratio, $F$:fidelity and $S_r$:solution ratio (explained in the text) over 10 runs at different numbers of shots $2^4$, $2^6$, $2^8$, $2^10$.

### 4.4.1 RUN1: Sequences

In RUN1 we picked 6 sequences on the $N = 6$ and $N = 11$ graphs, 3 sequences on each. This data were obtained from the simulations. We decided for sequences that would have a large solution ratio to see if the solution would be so evident also with the real noise from the machine.

We sum up the information on each sequence in Table 4.1, the details are explained in the caption. As you can see, the depth varies from 1 to 4.

The data gathered from the experiment is just the measured bitstring at 4 equally spaced time steps during the sequence of pulses. This was done to allow us to control what was happening during the run of the pulses. We plot the results in Figure 4.8 for all the sequences of Table 4.1. On the left we show the inverse approximation ratio $1 - R$ during the sequence: simulated with/without noise (continues/dashed lines) and from the experimental data (four dots). We see that the experimental data is in accordance with the simulated noisy signal but they both differ vastly from the simulated noiseless sequence.

The approximation ratio even exceeds 1, meaning that many states with positive energy are being measured, even though they are prohibited by Rydberg blockade. This means that either the condition is not respected or that noise is strongly affecting the signal. An effective, yet tragic, measure is to erase 50% of the measured bitstring that have higher energy, as measured with the classical Hamiltonian of the MIS problem (3.5). In this way we get rid of all (or most of) the states in which Rydberg blockade is violated, assuming it is due to noise both during the evolution and, in particular, at measurement. The effect of this discard are shown on the right column of Figure 4.8. We see that the refined data presents perfect agreement between simulation and experiment.

| ID | GRAPH | DEPTH | SHOTS | SEQ | RATIO |
|---|---|---|---|---|---|
| N1 | Diamond | 1 | 1028 | [250,520, 692] | 5.667 |
| N2 | Diamond | 2 | 1028 | [250,200, 200, 352, 800] | 14.833 |
| N3 | Diamond | 4 | 1028 | [250,200, 500, 444, 200, 200, 432, 416, 200] | 8.667 |
| N4 | Butterfly | 1 | 1028 | [250,492, 628] | 2.166 |
| N5 | Butterfly | 3 | 1028 | [250,428, 200, 648, 652, 200, 200] | 4.166 |
| N6 | Butterfly | 3 | 1028 | [250,200, 584, 200, 412, 232, 620] | 1.375 |

**Table 4.1: Sequences sent for RUN1** Each sequence of pulses alternates a global detuning pulse with $\delta = \pi$ and a global pulse with $\Omega = 2\pi$ MHz. SHOTS is the number of decided shot for the sequence, N QUBITS is the dimension of the graph, NOISE: if the sequence was obtained from noise simulations or not, PULSES: the sequence of pulses, RATIO: ratio of the measured solution MIS compared to the second most measured value.

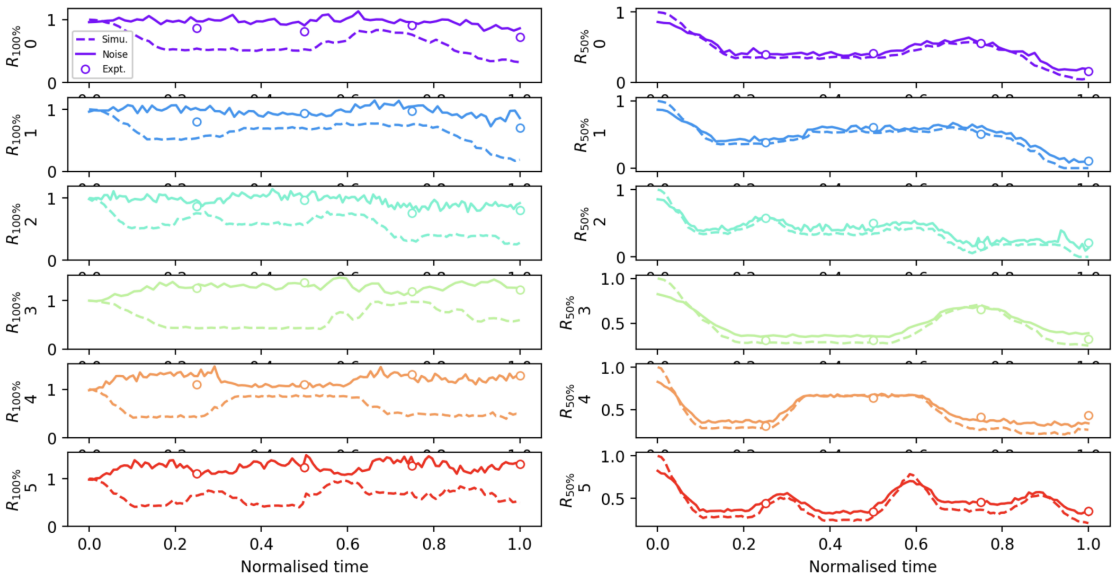



**Figure 4.8: Simulation vs Experiment: sequences** (left) Approximation ratio of the six sequences of Table 4.1. Dashed/continuous line is the simulation without/with noise, dots are the experimental data. (right) Same sequences data but discarding 50% of the bitstrings with largest energy.

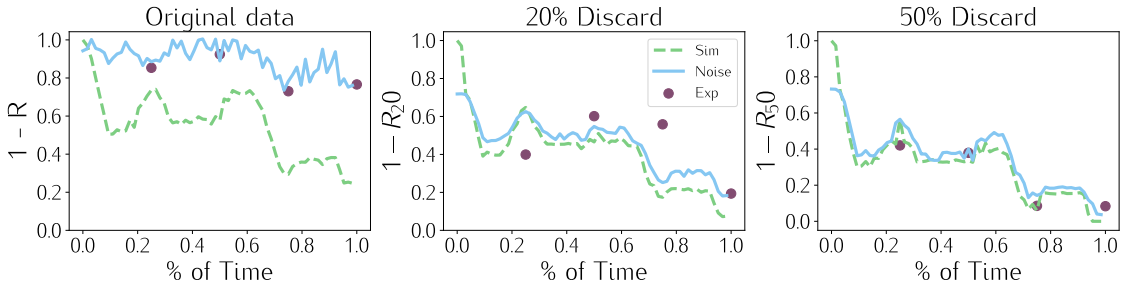



**Figure 4.9: Discard Analysis** (left) The approximation ratio $1 - R$ during the run of sequence $N3$. (middle) Same data but discarding the 20% most energetic bitstrings, (right) Same with 50%

Since a 50% means losing many measurements and we do not have many at disposal, we settled on a value of discard percentage to use with a qualitative analysis. This is shown in Figure 4.9 where we compare the results from the experiment (the four points for each sequence) with the simulation of the same sequence discarding 20% and 50% of the most energetic bitstrings. We ended up deciding for a 20% discard.

### 4.4.2 RUN 2: Loops

We ran full closed loops after implementing the discard percentage of 20% at each optimization step. The RUN2 part of the experiment comprehends 3 loops and its purpose is to prove that Bayesian Optimization is valid for this type of platforms. The properties of each loop are summed up in Table 4.2. We picked the graph with $N = 6$ and $N = 11$. Two loops were ran with 32 measurements per step and one with 64 measurements. This was done mainly for two reasons: avoid using too much computational time and testing the capabilities of Bayesian Optimization at low-shot rate. For this reason the 3 loops account for only 6 hours of computation.

| ID Loop | N | Total points | Actual points | SHOTS/step | TOT shots |
|---|---|---|---|---|---|
| L1 | 6 | 110 | 87 | 32 | 2784 |
| L2 | 6 | 110 | 96 | 64 | 6144 |
| L3 | 11 | 110 | 93 | 32 | 2976 |
| TOT | - | - | - | 11904 | |

**Table 4.2:** Parameters of the loops performed on the machine for RUN2.

During optimization we keep track of the relevant information about the quantum state at each step $|\psi(\boldsymbol{\theta})\rangle$. This includes the QAOA parameters presented in the previous section along with the Bayesian Optimization parameters. For the Bayesian Optimization we keep track of

- Kernel parameters: recall that the kernel with noise (defined in Chapter 3) is given by

$$k(\boldsymbol{\theta}, \boldsymbol{\theta}') = \sigma^2 \left(1 + \frac{\sqrt{3}||\boldsymbol{\theta} - \boldsymbol{\theta}'||_2}{\ell}\right) e^{-\frac{\sqrt{3}||\boldsymbol{\theta}-\boldsymbol{\theta}'||_2}{\ell}} + \sigma_N^2 \mathbb{I} \tag{4.7}$$

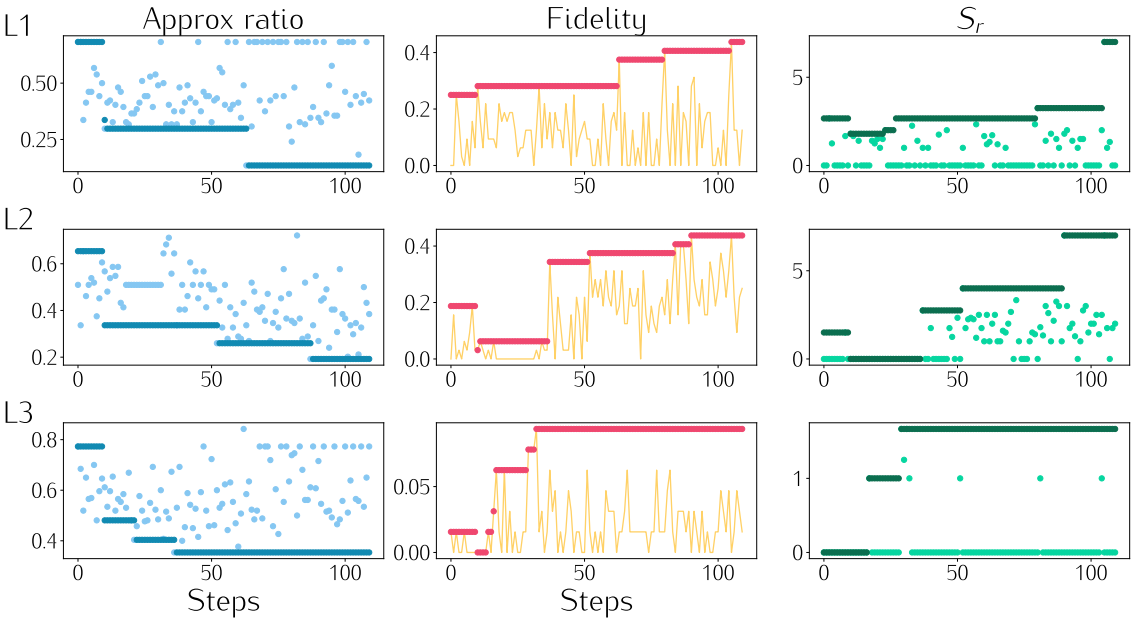



**Figure 4.10: Loops ran on the QPU with $N = 6$ and $N = 11$ shots: QAOA parameters**. Parameters during optimization for the loops L1, L2 and L3 described in Table 4.2. From left to right: approximation ratio, fidelity and solution ratio. The continuous line represent the best point obtained so far during optimization.

where

- constant $\sigma^2$ is parameter that sets the amplitude of the correlation between points,
- correlation length $\ell$ is a parameters that controls to which distance points are considered correlated by the Gaussian process,
- noise $\sigma_N^2$ is a parameter added to the kernel's diagonal to account for noise in the energy of the unknown function,

- $D_{i,i+1}$: distance between two consecutive sampled points, to check if the algorithm is under exploration of the landscape or exploitation (sampling points close to each other).

The approximation ratio is proportional to the parameter being optimized: the energy, and thus is the main parameter to be considered. By looking at the data obtained from the QPU during the loop in Figure 4.10 we see that the final approximation ratio reaches level around 0.2 for the loops on the $N = 6$ graph and about 0.4 for $N = 11$. This data alone does not give us much information, for this reason we plotted also the fidelity and the solution ratio as defined above.

For L1 and L2 we obtained large values for both energy and fidelity, with fidelities around 40% meaning a 40% chance to measure the solution bitstring and solution ratio $Sr \sim 7$ which means that the solution stands out in the state. The results are not so neat for L3 but we still obtain a fidelity around 8% and solution ratio $\sim 1.8$, the poorer quality is due to the dimension of the graph of course.

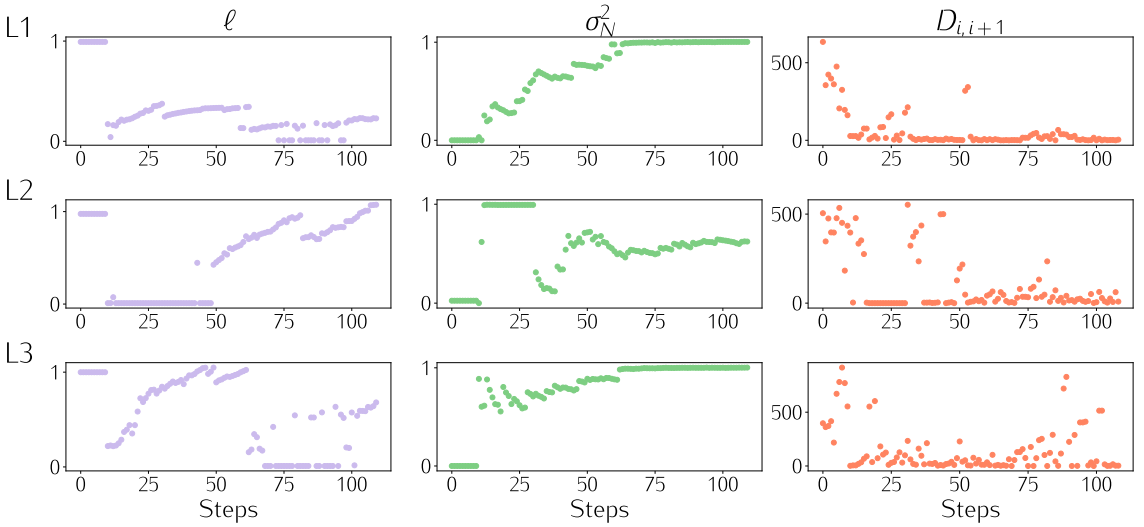



**Figure 4.11: Loops ran on the QPU with $N = 6$ and $N = 11$: Bayesian optimization parameters**. Parameters during optimization for the loops L1, L2a dn L3 described in Table 4.2. From left to right: $\ell$ the correlation length, $\sigma_N^2$ kernel noise and $D_{i,i+1}$ distance between consecutive points.

In Figure 4.11 we plot the parameters of the optimizer. These parameters help us interpret some of the results of the previous figure. For example the series of equal approximation ratio points at the beginning of L2 optimization are due to the fact that BO cannot find a suitable correlation length for the data obtained up to that point. The value remains in fact minimum for almost half of the optimization. This might be due to some noisier points sampled that brought large variations in the energy. The noise parameters is indeed stuck on the maximum value, signaling very noisy results and thus preventing optimization.

Overall, this first run with three loop allowed us to decide that the number of shots is probably too low to correctly perform optimization, considering also the discard of 20% of the bitstrings.

### 4.4.3 RUN3: Final loop

By combining the knowledge gained with the first two runs we finally ran one last optimization loop on an even bigger graph of $N = 15$ qubits shown in Figure 4.12 at depth $p = 5$. The optimization was successful, in the same picture we show the final state that was sampled and where the solution bitstring was measured more often than all the other ones.

The loop was run using all the remaining shots which were in the order of 50 000. We decided to run 100 steps of optimization with 20 initial points of training at 450 shots per step. Considering the discard of 20% we are optimizing with 360 effective shots. We summed up the results in Figure 4.13 where we show only the relevant parameters. The plots show a comparison between the optimization on the QPU and a simulation of the same loop.

The parameters show us a well performed optimization loop. We can in fact agree

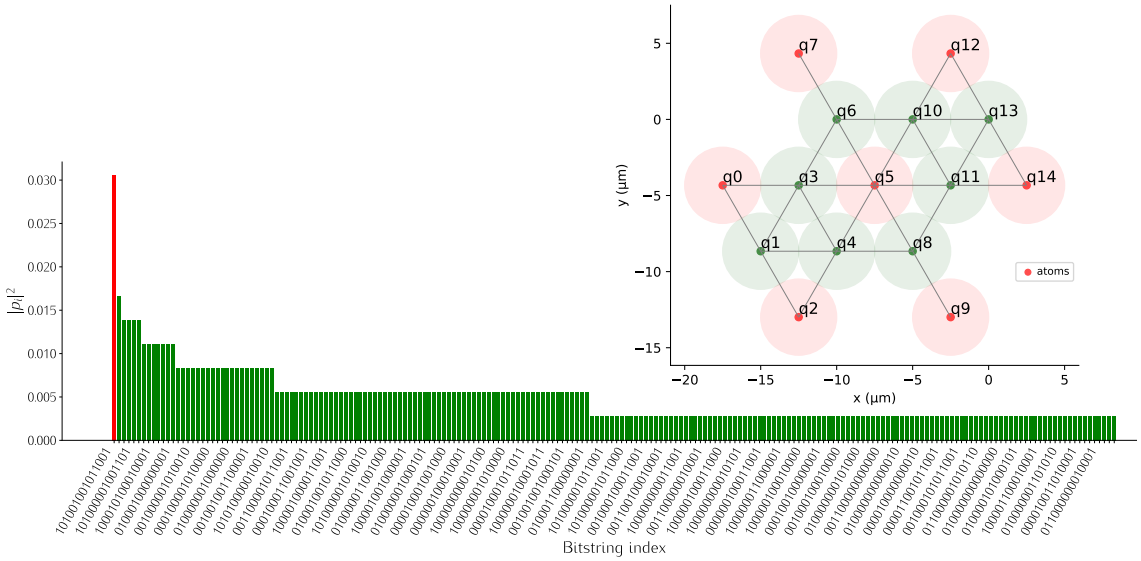



**Figure 4.12: Solution state with graph** The graph of $N = 15$ qubits used for the experiment plotted on the register with the red qubits highlighting the MIS solution state that we are looking for corresponding to the bitstring

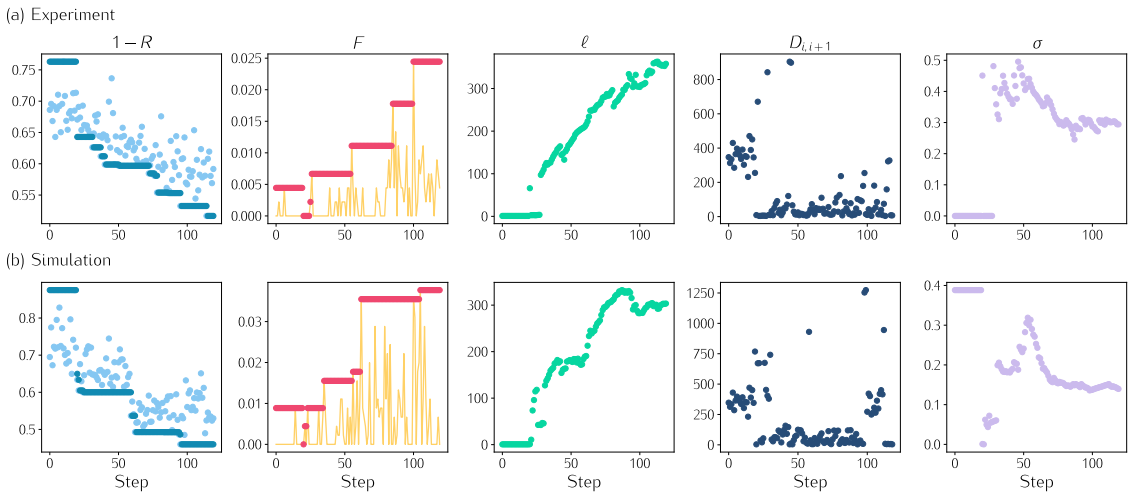



**Figure 4.13: Optimization ran on the real QPU on a 15 qubits graph.** (a) Parameters during optimization on the QPU. (b) Simulation of the optimization on the same graph.

that the approximation ratio descends practically monotonically and the fidelity improves multiple times. On the optimizer side, the correlation length keep rising, meaning that BO keeps exploring the landscape for better points at each step, without necessarily stopping on one. This is seen also on the distance $D_{i,i+1}$ between consecutive points that alternates larger distances in the order of the hundreds with few local distances in the order of the tens. This pattern is exactly typical of Bayesian Optimization that concentrates on an area and if not satisfied jumps to another part of the landscape.

The same characteristics can be noted also on the simulated loop. The main difference between the two is the noise level, we can see that in both scenario it settles around a specific value which is double in the real scenario compared to the simulation. This means that even if we accounted for noise in the simulation the real source of noise in the QPU was larger and thus affected more strongly the results.

We can say with confidence that we would have gotten even better results with more steps of optimization, this is what the trend in the parameters clearly indicates. Yet, a solution ratio of 1.8 was reached, which is not trivial given the dimension of the graph and the limited number of runs available.

# 5 | Measurement Based Quantum Computation

In this chapter we want to show an approach to quantum computation with Rydberg atoms that deviates from the standard analog or digital approaches because it is based on projective measurements: Measurement Based Quantum Computation (MBQC). We will start by introducing MBQC in its original form, that is the cluster state formulation and then present the correlation space approach that was developed to exploit states different from the cluster state. The latter is the premise of the work developed by [Cré22] in which they showed that a dimer state made on Rydberg atoms can be used as resource for quantum computation using only a specific set of measurements that are currently available on state-of-the-art Rydberg atoms' platforms.

Our purpose is to show the resources needed to perform MBQC on the dimer state on a neutral atom's platform. Therefore, we will show the basic functioning of this approach taken from [Cré22] and then add our contribution which consists in improving the scheme and produce an estimation of the resources needed for a generic algorithm like Deutsch-Jozsa. The research in this chapter sums up the work that I brought on during a six-months internship at the company PASQAL.

## 5.1 Cluster state quantum computation

Measurement Based Quantum Computation (MBQC) is a framework for computation using only an initial entangled state and projective measurements. It was first developed by Raussendorf and Briegel under the name of *1-Way Quantum Computer* [RB01b, RBB01] because the measurements are performed with a specific order creating computation along lines of qubits in one specific space direction.

This computation happens in three steps. First, given a graph of $N$ qubits, a cluster state [BR01] is created by applying $CZ$ gate to each qubit pair connected by an edge, $|\Phi\rangle = \prod_e CZ_e \, |{+}{+}\rangle^{\otimes N}$. The cluster state is part of the more general family of graph states [RB01a] and it has strong entanglement requirements [BR01] that makes it hard to create even though we have some small experimental realizations, including with neutral atoms [MGW+03, BLS+22]. Secondly, consecutive one-qubit measurement on different basis are applied to qubits. Finally, the correction of possible errors arising from this computation is performed.

An enlightening example for this type of computation is the teleportation a quantum state from one qubit to another. Let us call $|\Phi\rangle_\psi$ the cluster state created with one qubit in a state $\psi = \alpha|0\rangle + \beta|1\rangle$ and the second one in the $|+\rangle$ state, that is after being entangled the system is in the state:

$$|\Phi\rangle_\psi = \alpha|0+\rangle + \beta|1-\rangle \tag{5.1}$$

Now, we measure along $X$ the first qubit and let us assume we obtain an eigenvalue $x = 0/1$ if the qubit ends up on $|\pm\rangle$. This entails that the state was projected by a projector $P_{\pm x}$ like so:

$$\begin{aligned} P_{\pm x}|\phi\rangle_\psi &= |\pm\rangle\langle\pm|\left[\alpha|0+\rangle + \beta|1-\rangle\right] \\ &= \alpha|{+}{+}\rangle \pm \beta|{+}{-}\rangle \\ &= |\pm\rangle\left[\alpha|+\rangle \pm \beta|-\rangle\right] \\ &= |\pm\rangle HZ^x|\psi\rangle \end{aligned} \tag{5.2}$$

The result is that we are able to transport the information of a qubit, i.e. its state, along a chain of entangled qubits only by performing measurements.

**Qubit rotations** The intuition between the previous example can be extended into operations on qubits and creating entanglements between them. It is pretty straightforward to show that any single qubit gate $\in SU(2)$ can be applied to a specific qubit by using a total of 5 qubits and 4 measurements. In fact, let us assume we can measure qubits along the basis:

$$B_j(\varphi) = \left\{\frac{|0\rangle + e^{i\varphi}|1\rangle}{\sqrt{2}}, \quad \frac{|0\rangle - e^{i\varphi}|1\rangle}{\sqrt{2}}\right\} \tag{5.3}$$

where $j$ is the qubit on which you apply the measurement on and we consider outcome $s_j = 0, 1$ if the first/second state is measured. To apply a generic rotation $U_R \in SU(2)$ we can decompose it in its Euler angles:

$$U_R(\alpha) = U_R(\varepsilon, \eta, \xi) = U_x(\xi)U_z(\eta)U_x(\varepsilon) \tag{5.4}$$

and we create the initial state by applying the entangling operation of the cluster state to the state $|\psi + + + +\rangle$ where $|\psi\rangle$ is the state on which we wish to act with $U_R(\alpha)$. After this we measure:

1. $B_1(0)$ (which is X on the first qubit)
2. $B_2\left((-1)^{s_1+1}\varepsilon\right)$
3. $B_3\left((-1)^{s_2}\eta\right)$
4. $B_4\left((-1)^{s_1+s_3}\xi\right)$

and the final qubit is found in the state

$$U_\Sigma U_R(\alpha)|\psi\rangle \quad \text{with} \quad U_\Sigma = \sigma_x^{s_2+s_4}\sigma_z^{s_1+s_3} \tag{5.5}$$

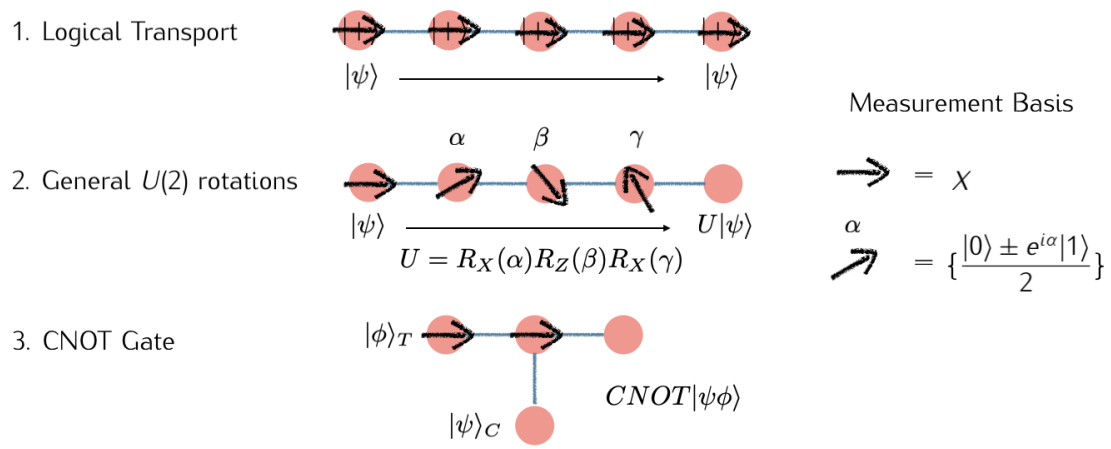



**Figure 5.1: 1-Way QC**. Summary of the three operations that allow MBQC to be universal. Logical transport means being able to transport the quantum state of a qubit to another (teleportation). A general $U(2)$ rotation along with CNOT are then sufficient to perform any algorithm.

**CNOT gate** A CNOT gate, to reproduce entanglement, requires only 4 qubits and 2 measurements along $X$ [Tab11]. By combining any $U_R(\alpha)$ with a CNOT any circuit can be represented, in this way the so called "1-Way Quantum Computation" scheme was proven to be universal for quantum computation. A summary of this operations is shown in Figure 5.1.

**Pros and Cons** Two advantages of this method are the fact that all Clifford gates can be applied virtually at the same time and it allows us to rethink quantum computation beyond the digital basis approach, for example forming gates that have no counterpart, like the famous bit-reversal gate [RBB01]. Since its invention in the early 2000s many ideas and research field were born stemming from to this method. To cite a few, the proof of its equivalency to another teleportation-based approach [CLN05], the development of Measurement Calculus [DKP07] to generalize and simplify the languages of quantum computation with the 1-Way computer, the invention of $ZX$-calculus and finally a Measurement Based Quantum Error Correction scheme [RHG06].

The drawbacks of this method is that it starts from a cluster state which is complicated to create experimentally. The first realization [Wei21] of a cluster state was done by Mandel et al. in the Bloch group [MGW+03] using collision between atoms but the technology of local addressing was not ideal for the task. Another preparation of a cluster state with Rydberg atoms was performed by Lukin et al. [BLS+22]. In this implementation they are able to transport qubits maintaining entanglement efficiently, thus they prepare the 1D cluster state on 12 qubits. Most efforts have not improved much more on this and the research on the topic has been focusing mostly on the theoretical aspects. These technical difficulties favoured the developments of other approaches like the one we see in the next section and protagonist of our research.

## 5.2 Correlation space approach

Inspired by the tensor-network interpretation of MBQC [VC04] a novel schemes, called the correlation space approach [GE07, GESPG07], was proposed. This approach stems from the search of resource states that can emerge as groundstates of local Hamiltonians (the cluster state arises from a 5-body interacting Hamiltonian [Nie03]). Differently from cluster-state (or 1-Way) MBQC – in which the physical qubits used for measurements are also the qubits of computation – this new approach uses physical qubits for measurements but the computation happens in the virtual Hilbert space of their tensor-network representation. This new approach was relevant because it can be proven that other states beyond the cluster state are universal for quantum computation through measurements only.

The recipe for correlation space MBQC (CS-MBQC) is simple:

1. we write the state in its PEPS/tensor-network representation
2. the boundary vectors in the virtual space are interpreted as qubit states
3. the measurements on the physical sites of the resource state induce unitary operations in the virtual space on the boundary vectors.

Understanding this topic requires knowledge of tensor networks and matrix product state (MPS) representations. Their explanation is outside the scope of this work so we will briefly introduce the topic with the use of examples, for a complete introduction see [CPGSV21].

**MPS representation** The purpose of an MPS is to write each coefficient of a quantum state of a many-body system like this:

$$|\Psi\rangle = \sum_{s_1,s_2,\dots s_N=0}^{d-1} \alpha_{s_1,s_2,\dots,s_N} |s_1, s_2, \dots, s_N\rangle \tag{5.6}$$

into contractions of matrices:

$$|\Psi\rangle = \sum_{s_1,s_2,\dots s_N=0}^{d-1} \langle R| A_N[s_N] \dots A_1[s_1] |L\rangle |s_1, s_2, \dots s_N\rangle \tag{5.7}$$

In equations (5.6), (5.7) the state $|s_1, s_2, \dots, s_N\rangle$ represents a chain of $N$ sites, each with $d$ degrees of freedom. $A_i[s_i]$ is a set of $d$ matrices defined for each site $i$ and $|R\rangle, |L\rangle$ are two boundaries vector used to perform the contraction. By equaling the two equations we obtain

$$\alpha_{s_1,s_2,\dots,s_N} = \langle R| A_N[s_N] \dots A_1[s_1] |L\rangle \tag{5.8}$$

which sums up everything we need to know about MPS. It is a set of matrices (plus 2 boundary vectors) constructed specifically so that their $d^N$ possible contractions correspond to the $d^N$ coefficient of the wavefunction $|\Psi\rangle$.

Two examples might be useful:

- GHZ state. By inspection we can check that the $N$ qubits $GHZ$ state:

$$|GHZ\rangle = \frac{1}{\sqrt{d^N}} \sum_{i=0}^{d-1} |i, i, \ldots, i\rangle , \tag{5.9}$$

can be put in MPS form by taking matrices with dimension $D = 2$ which is called the bond dimension:

$$A[0] = \frac{1}{\sqrt{d}} \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} = |0\rangle\langle 0| , \quad A[1] = \frac{1}{\sqrt{d}} \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} = |1\rangle\langle 1| \tag{5.10}$$

It is easily interpretable because $A[0]A[1] = 0$, thus the only two contractions which give a nonzero value are $A[i]A[i] \ldots A[i]$.

- The W state. The state

$$|W\rangle = |100\ldots\rangle + |010\ldots\rangle + \cdots + |0\ldots001\rangle \tag{5.11}$$

can be cast in MPS form with

$$A[0] = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = |0\rangle\langle 0| + |1\rangle\langle 1| , \quad A[1] = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} = |0\rangle\langle 1| \tag{5.12}$$

This is also easy to interpret, if the state of a qubit is 0, its matrix is the identity so it has no impact in the contraction of all the matrices, whereas $A[1]$ acts on $|1\rangle$ and flip it to $|0\rangle$. Thus using $|R\rangle = |0\rangle$ and $|L\rangle = |1\rangle$ we get that any combination of qubits with more than one "1" becomes null and the one with only one "1" are contracted to $\langle R| A[1] |L\rangle = 1$.

**AKLT as resource state** The celebrated AKLT state [AKLT87] is the groundstate of the bilinear biquadratic spin-1 chain and was the starting point for the development of MPS representations. We will show directly on this state how the MPS representation allows for quantum computation in the correlation space. For this state we have 3 physical degrees of freedom at each site corresponding to a spin 1 particle with spins $0, 1, 2$. It is represented by an MPS with matrices of dimension $2 \times 2$ such that:

$$\begin{aligned} A[0] &= H \\ A[1] &= \frac{1}{\sqrt{2}} |0\rangle\langle 1| \\ A[2] &= \frac{1}{\sqrt{2}} |1\rangle\langle 0| \end{aligned}$$

with boundary vectors:

$$|L\rangle = |R\rangle = |0\rangle \tag{5.13}$$

Now we measure the first site from the right in the spin basis and we might obtain $0, 1$ or $2$. Let us say we measured $s = 0$, this means that in the MPS the matrix of the first site collapses to $A_1[0] = H$. Since the right boundary vector is $|R\rangle = |0\rangle$ we have that the MPS representation of this chain transforms as:

$$\langle L| A_N[s_N] \ldots A_1[s_1] |R\rangle \to \langle L| A_N[s_N] \ldots H |R\rangle = \langle L| A_N[s_N] \ldots |+\rangle \tag{5.14}$$

which means that we now have a new boundary right-vector that switched from the $|0\rangle$ state to the $|+\rangle$ because it received the action of the gate $H$, meaning the contraction with the matrix $A_1[0]$.

This simple result tells us that in the space of the MPS the boundary vector is acted upon with unitary transformation and thus is like a qubit that evolves under a quantum circuit. This correspondence is due to the fact that even if the physical system has 3 degrees of freedom on each site, since the bond dimension is 2 in the correlation space we have a logical qubit undergoing computation.

This reasoning can be extended to other types of measurements [GE07]. Let us define measurement along a generic direction $\phi$ on the plane spanned by $|1\rangle$ and $|2\rangle$ as the projection onto the basis states $|\pm\phi\rangle = \{\frac{1}{\sqrt{2}}|1\rangle \pm e^{i\phi}|2\rangle\}$ then by definition the matrix of the site collapses to:

$$A[\pm\phi] = \langle\phi|1\rangle A[1] \pm \langle\phi|2\rangle A[2] = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & e^{i\phi}\end{bmatrix} = S(\phi)H \tag{5.15}$$

which is equivalent to applying an Hadamard gate followed by a phase gate $S(\phi)$. In this way it is proven straightforwardly that any rotation of $SU(2)$ can be performed on a logical qubit [GESPG07].

### 5.2.1 CS-MBQC on Rydberg Atoms

State-of-the-art platforms of Rydberg atoms can perform projections on multiple qubits by moving them from the simulation plane to a computation plane and shining and measuring the qubits before placing them back into the simulation plane [BLdL+18]. This allows for three types of measurements (for more details see [Cré22]) that we will use to build a framework of MBQC with Rydberg atoms:

- Measurement (I). An ensemble of two neighbouring atoms is represented by the basis states:

  where the green dots means that the atom is in the Rydberg state. If we evolve the two atoms for time $\tau = \pi/\Delta$ with $\Delta/\Omega = \sqrt{2}/3$ and we measure on the Rydberg basis, the state is projected onto:

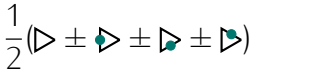
  

  , depending on the outcome of the measurement.

- Measurement (II). An ensemble of three atoms has basis:

  If we evolve them for $\tau = 4\pi/(3\sqrt{3}\Omega)$ with $\Delta = 0$ they are projected into [VLV21]:

  (5.16)

  Where the signs depend on the measurements' outcomes [Cré22].

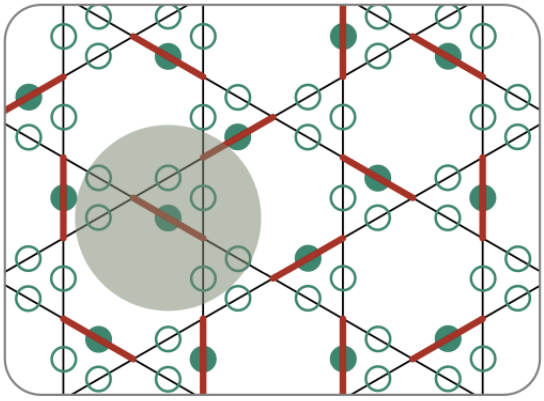
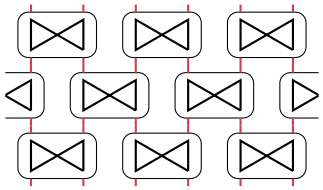


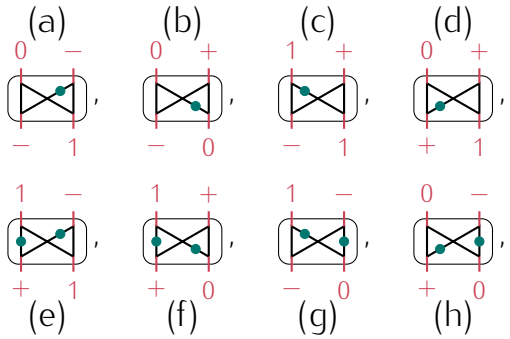
**Figure 5.2:** Dimer state on the Kagome lattice with Rydberg atoms. On each edge of the Kagome lattice we place an atom, in this way they form the Ruby lattice. If an atom is on the Rydberg state (green dot) we consider that as a dimer. Credit: [Cré22].

- Measurement (III). Involving only one atom: a far detuned laser $\Delta >> \Omega$ applied for a time $\tau$ which applies a phase of $\phi = \Delta t$ to the excited state $|e\rangle$.

A recent work [Cré22] proposed how to implement correlation space MBQC on a groundstate made with Rydberg atoms. The resource state considered is the dimer state on the Kagome lattice [MSP02] which is shown in Fig. 5.2. A dimer is any edge that connects to excited vertices on a lattice whereas a covering is the state formed by all the dimers arranged so that every vertex is touched by exactly one dimer. Thus, a dimer state is the superposition of all possible dimer coverings of a lattice.

On a Rydberg machine we can place the atoms on the edges of the Kagome lattice (which form the Ruby lattice) and consider a dimer being signaled by and excited state (Fig. 5.2). In this case the dimer state is the superposition of all coverings of the Ruby lattice. To understand the method proposed in [Cré22] we start by writing the PEPS representation of the dimer state:

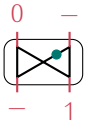
$|\Phi\rangle =$  , (5.17)




(5.18)

The state in (5.17) is created by grouping atoms in group of 6 forming a bow-ties of 6 physical qubits. Each of them has $2^6$ possible states with only 8 of them being physically possible due to the Rydberg constraint on the Kagome lattice. We show the allowed states configurations in (5.18).

For each allowed state we define a tensor with bond dimension 4. These tensors can be chosen freely with the constraint that the contraction of all the tensors needs to always return a dimer covering. Exploiting this freedom, we create this tensors as operators acting on pairs of qubits, as it is customary in the correlation space approach [GESPG07, GE07]. For example, tensor (5.18)(a) is chosen like so:



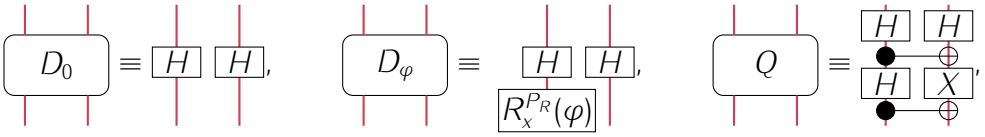

$$= |-\rangle\,|1\rangle\,\langle -|\,\langle 0| \tag{5.19}$$

and can be interpreted as an operator acting on two logical qubits from top to bottom. It can be checked directly that the choice made in (5.18) is such that the only combinations of tensors that do not contract to zero are dimer coverings.

Following this line of reasoning, we can interpret the red wires in (5.17) as logical qubits and the consecutive measurements from top to bottom as quantum circuit operations. To show that universal quantum computation is possible under this premises, in [Cré22] they create three sequences of measurements, combining measurement (I), (II) and (III) above. Each of these sequences projects a single bow-tie into a superposition of states, and thus tensors in the PEPS space, which correspond to these three operations:

- Decoupling gate $D_0$: it performs $H$ on both logical qubit;
- Rotation gate $D_\varphi$: it performs $R_x(\varphi)$ to either the right or left qubit with a success rate of 50% depending if we measure $P_R = 0$ or 1;
- Entangling gate $Q$ between the two qubits

Following the notation of [Cré22], the action of each of the three set of measurements can be represented as quantum gates acting on the left, right or both computational wires:




(5.20)

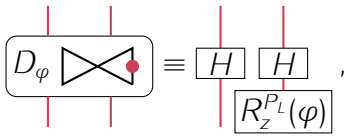
Details on these gates and how they are created can be found in Appendix B. The relevant detail is that the rotation gate $D_\phi$ is applied only when the measurement outcome is $P_R = 1$ as previously stated.

## 5.3 Extension of the model

We now present our contribution to this work. In the first part of this section we show possibilities and limits of this scheme by acting at the pulses and measurements level. In the following part we show all the conventional and original gates that can be created.

### 5.3.1 More measurement gates

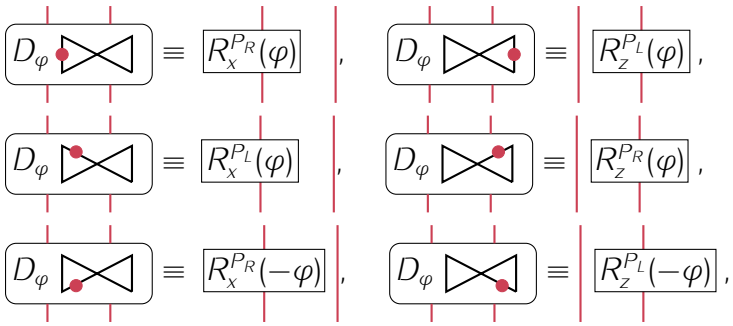
**Creation of a deterministic gate** The $D_\varphi$ gate has the issue of not being deterministic. Let us start by applying the sequence of this gate with the exception of dephasing by $\varphi$ the right-most qubit in the bow-tie instead of the left-most one. Contrarily to what stated in [Cré22] we do not obtain $R_x(\varphi)$ on the right but $R_z(\varphi)$ instead:




(5.21)

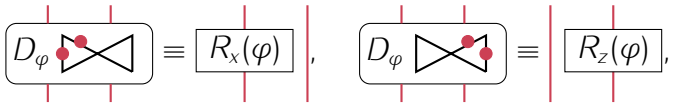
In which we renamed the gate in order to make explicit the qubit that receives the $\phi$ phase. The gate $R_z$, similarly to $R_x$, allows for universal quantum computation since $R_z(\pi/4)$, also known as the $T$ gate, is non-Clifford. In the same fashion as before the gate is dependent on the measurement of $P_L$ and it is thus effective only 50% of the time.

We show now all the possible single qubit rotations that can arise according to the qubit that is dephased. For the sake of simplicity we do not show the initial $H$ gates that are given for granted:




(5.22)

The fact that some configurations (such as number 1 and 5) produce the same rotation on the same qubit but depending on $P_R$ or $P_L$ hints that there might be a way to produce a deterministic gate. That is indeed the case, by direct calculation we are in fact able to obtain the following, deterministic gates:

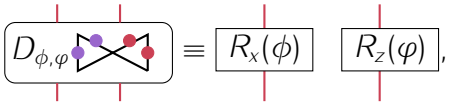



(5.23)

As specified in the last equation this solves the problem of the stochastic gates since measuring either $P_R$ or $P_L$ will always lead to the same gate being applied. By combining other pairs of Eq. (5.22) we would always encounter a non-deterministic output: the application of the gate or the correct angle phases. Therefore, we discard them.

We can also extend the same reasoning to combination of 3, 4, 5 or all the gates in Eq. (5.22). First of all, it can be noticed that any odd number of combinations will always produce uncertainty. That means that either you end up with a probabilistic phase $(-)^{P_R/P_L}$ or gate $R_{x/z}^{P_R/P_L}(\varphi)$. Second of all, if we consider dephasing every qubit and thus taking a superposition of all the gates would cancel the action of either

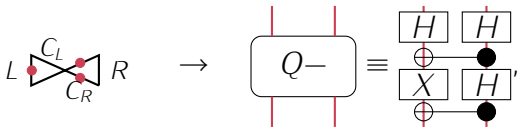
$R_x$ or $R_z$ according to $P_L$ or $P_R$. So this case it is not interesting. Finally, we can build a quadruplet by combining the pairs of Eq. (5.23), in particular we can rotate the two logical qubits by different angles at the same time, in fact:




(5.24)

where we defined the gate as $D_{\phi,\varphi}$ as an extension of the previous $D_\varphi$ and where the purple dots represent a phase of $\phi$ and the red ones of $\varphi$, respectively.

**More on the entangling gate** We can analyze further the potential of also the entangling gate $Q$. First, it is easy to show that gate $Q$ can be implemented in an identical version with the control and target qubits inverted. We call this gate $Q-$ and it can be derived by dephasing a different set of initial qubits. That is, we follow the same steps in Appendix B but we start by applying a phase $\pi$ to the qubits as shown in the left side of Eq. 5.25, and obtain the gate:




(5.25)

This gate will turn out useful for the definition of a new operation in the next Section.

**Non-unitarity of other gates** It is important to notice that in order to apply a set of gates on the logical qubit, i.e. a unitary matrix, the scheme allows for only two options. The first one is to produce a superposition of 4 of the 8 tensors (5.18). In order for this superposition to represent a unitary operation on the logical space, each of the possible combinations of incoming qubits needs to appear once. That means that the allowed superpositions include tensors (5.18)(c,d,g,h) or tensors (5.18)(a,b,e,f) (cf. the derivation of $D_0$ and $D_\phi$ in Appendix B). In this case, we can only act with local gates on the logical qubits (such as $H$ in the $D_0$ bow-tie and rotations in the $R_x$ bow-tie).

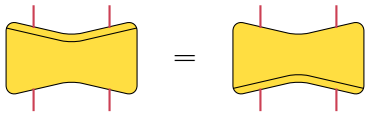
The second option is to produce a superposition of all the 8 tensors (5.18). In order for this operation to act as a unitary on the correlation space the phases need to be adjusted accordingly as shown for example in Eq. (5.25) (a simple superposition without specific phases would not create a unitary matrix). In this way, entanglement between the two logical qubits is created.

It can be checked directly that any other superposition of tensors will not produce a unitary operation on the incoming logical qubits. This reduces the number of significant combinations of measurements we can perform and concludes our discussion on improvements.

### 5.3.2 Rewriting rules and additional gates

In this part we introduce a new graphical representation of gates that allows for easier comprehension of the algorithms constructed in the next section. We also show how to construct some more elaborated gates by exploiting the nature of MBQC in this framework.

**Decoupling gate** The decoupling gate $D_0$ acts almost trivially on the logical space qubits. To allow easier comprehension of this scheme we introduce a new graphical visualization of this gate:

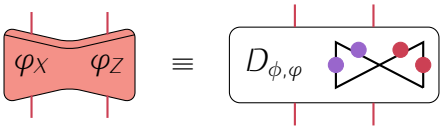


(5.26)

The convexity of the top and bottom edges indicates which gates can be concatenated with this one. That is, only gate with concave edges. In fact two decoupling gates cannot be applied consecutively to the same logical qubits. The double edge means where the $H$ gates are positioned and clearly in does not matter for the $D_0$ gate.

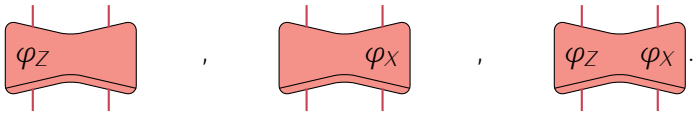
**Single Qubit rotations** We showed in the previous section how to perform a rotation of a logical qubit with the set of measurements $D_\varphi$. With one bow-tie we can perform the rotations $R_x$ to the left-most qubit and/or $R_z$ to the right-most. Following the new notation we define the gates:


(5.27)



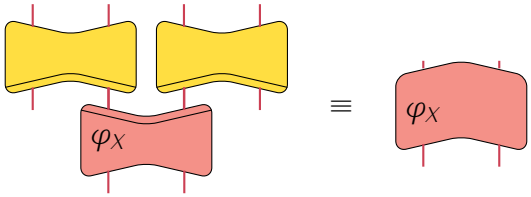

(5.28)

which clearly state which rotation is happening on which qubit. Thanks to the permutation rules $HR_x = R_zH$ we can exchange the action of the $R_x$ and $R_z$ by switching the $H$ gates at the bottom. This creates the following three gates:




(5.29)

Consider now that you want to act on the left qubit with a $R_x$ rotation without having $H$ gates to deal with. This can be done by inserting two decoupling gate before the rotation gate and this gives a new rule:



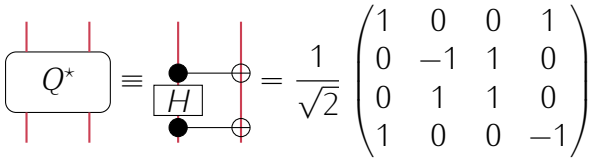

(5.30)

The gate on the r.h.s. of this equation has two important features: the double edge is gone because the $H$ gates of the l.h.s. simplified and the top is convex, signifying that it can only be preceded by a concave gate. Finally, in order to apply $R_y$ we can exploit the relations:

$$R_y(\varphi) = R_z\big(-\frac{\pi}{2}\big)R_x(\varphi)R_z\big(\frac{\pi}{2}\big) \tag{5.31}$$

$$R_y(\varphi) = R_x\big(-\frac{\pi}{2}\big)R_z(-\varphi)R_x\big(\frac{\pi}{2}\big) \tag{5.32}$$

**CNOT gate** To obtain a CNOT from the entangling gate $Q$ of our MBQC scheme, we get rid of the initial two $H$ gates (this can be done by acting with the decoupling gate on the left and right qubits). Then, we call the resulting gate $Q^\star$:




$$\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 0 & 0 & 1\\ 0 & -1 & 1 & 0\\ 0 & 1 & 1 & 0\\ 1 & 0 & 0 & -1\end{pmatrix} \tag{5.33}$$

in which we neglected momentarily the $X$ on the second qubit and explicited its matrix form. We can see that the gate $Q^\star$ acts a the transformation from the digital basis to the Bell basis $U^{\text{st}\to\text{Bell}}$. Interestingly, in [BdV08] they work out a decomposition of $U^{\text{st}\to\text{Bell}}$ that allows us to go from $Q^\star$ to CNOT. In fact, the following equivalences hold:

$$\begin{aligned} Q^\star = U^{\text{st}\to\text{Bell}} &= R_z^{(1)}(-\frac{\pi}{2})\sqrt{\text{SWAP}}R_x^{(1)}(\pi)\sqrt{\text{SWAP}}R_y^{(1)}(\pi)R_z^{(1)}(\frac{\pi}{2}) \\ CNOT = R_y^{(1)}(-\frac{\pi}{2})R_x^{(1)}(-\frac{\pi}{2})R_x^{(2)}(\frac{\pi}{2})&\sqrt{\text{SWAP}}R_x^{(1)}(\pi)\sqrt{\text{SWAP}}R_y^{(1)}(\frac{\pi}{2}) \end{aligned} \tag{5.34}$$

in which the superscript represent the qubit on which the rotation acts, whereas the $\sqrt{\text{SWAP}}$ necessarily acts on both qubits. By direct inspection of Eqs. (5.34) we find the relation connecting $Q^\star$ to CNOT is:

$$CNOT = K_1 Q^\star K_2 \tag{5.35}$$

where $K_1$ and $K_2$ are a set of single qubit rotations:

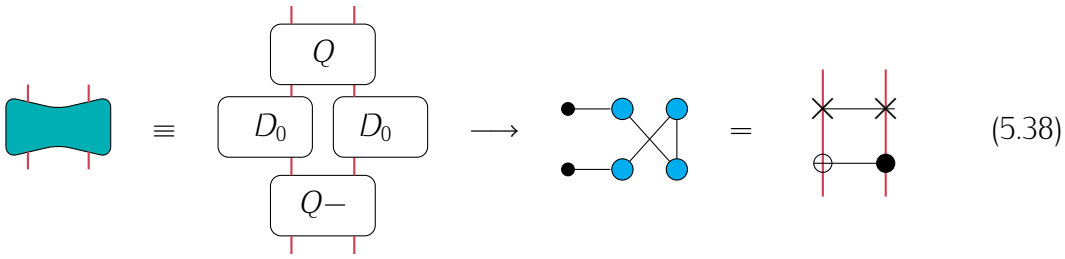
$$K_1 = R_x^{(1)}(-\frac{\pi}{2})R_x^{(2)}(\frac{\pi}{2}) \tag{5.36}$$

$$K_2 = R_z^{(1)}(\frac{\pi}{2})R_y^{(1)}(-\frac{\pi}{2}) \tag{5.37}$$

**SWAP gate** A SWAP gate has a simple decomposition in terms of three consecutive CNOT gates in which the middle one has the target and control qubits inverted. This means that given the sequence that implements a CNOT shown above we only need to repeat it three times adding two decoupling gates in the middle.

**SWAP-CNOT (SNOT) gate** Our framework, as much as MBQC in general, invites us to go beyond the standard approach of digital computation. That is, it is not always convenient to reproduce an algorithm by reproducing its elementary gates with the MBQC scheme while rather utilize the MBQC to create new types of operations on qubits. That is exactly the idea behind the qubit-reversal gate presented at the beginning.

The first non–trivial composition of gates that we can create is the consecutive application of $Q$ and and $Q^-$, separated by a decoupling gate $D_0$ (necessary due to the topology of the bow-ties). This operator can be investigated with $ZX$-calculus [CD07]. Using the package `pyZX` [KW20], we can rewrite it as the following circuit, which is much simpler and in fact is a SWAP-CNOT:




(5.38)

The arrow means that the gates on the left were translated into $ZX$-calculus and fed to the optimizer of `pyZX` to produce the simplified version which corresponds simply to a SWAP-CNOT.

### 5.3.3 Resource estimation of Algorithms

We now use the gates defined in the previous section to show how to implement quantum algorithms. Firstly, we will show how measurements need to be combined in order to obtain the desired gates that build up a digital computation algorithm and then we will estimate the resources needed to perform Deutsch-Jozsa and Grover on a Rydberg platform. We now define what we mean by resources:

1. Number of atoms
2. Number of pulses
3. Types of pulses. We divide pulses into driving pulses, the ones applied before measurements and change of basis pulses, the ones applied to perform measurement on other basis. The latter ones always come in pairs. The driving pulses are 3 in total, corresponding to the three measurements described before. The change of basis in only one, used in the gates $D$ and $G_\phi$ to measure in the $|g\rangle \pm |e\rangle$ basis.
4. Types of measurements. Since we are allowed to measure only on the digital basis by fluorescence we consider only one type of measurement and we will omit it in the estimation.
5. Number of measurements. The number of fluorescence measurements

In these terms, we can sum up the resources needed to perform the three types of measurements described in the previous section. The requirements by each gate are summed up in table 5.1.

#### Deutsch–Josza

Deutsch–Jozsa (DJ) is a simple quantum algorithm developed to show an example of quantum speedup. We have a binary function $f : \{0,1\}^N \to \{0,1\}$ that is either balanced (half the bitstrings evaluate to 0 and half to 1) or constant (all bitstrings evaluate to 0/1). The DJ algorithm understands if the function is balanced or constant with only one run of the circuit. Figure 5.3 shows the DJ algorithm in its balanced version. That is, if all the qubits are measured on the $|0\rangle$ state, the function is balanced otherwise it is constant. It requires one ancilla qubit ($q_3$ in Fig. 5.3) that is entangled with all the computational qubits and then discarded.

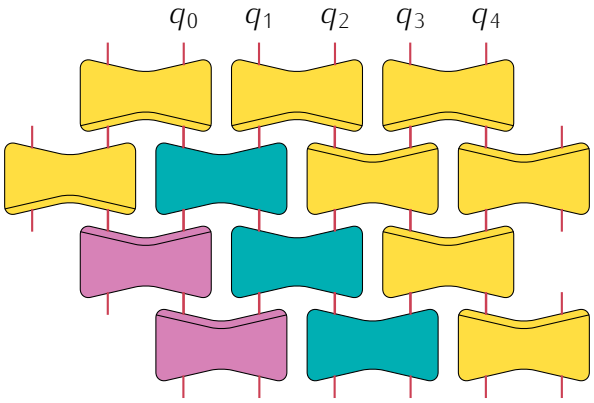
| Gate | Atoms | Meas. no. | Change of basis Pulses no. | Driving pulses no. |
|---|---|---|---|---|
| $D_0$ | 6 | 4 | 4 | 4 |
| $D_\varphi$ | 6 | 4 | 4 | 6 |
| $DD_\varphi$ | 6 | 4 | 4 | 8 |
| $Q$ | 6 | 2 | 0 | 5 |
| $M$ | 6 | 1 | 0 | 0 |

**Table 5.1:** Resources needed per gate. $D_0$ applies $H$ to both qubits, $D_\varphi$ rotate the left/right one by $R_{x/z}(\varphi)$, $D_{\phi,\varphi}$ rotates the left one by $R_x(\phi)$ and the right one by $R_z(\varphi)$, $Q$ entangles the qubits, $M$ is the measurement gate.

| Gate | Atoms | Meas. no. | Change of basis Pulses no. | Driving pulses no. |
|---|---|---|---|---|
| CNOT | 30 | 18 | 16 | 29 |
| SWAP | 42 | 26 | 34 | 33 |
| S-NOT | 24 | 12 | 8 | 18 |

**Table 5.2:** Resources needed to run common gates in digital computation.

The only non-trivial part of the circuit is the series of $CZ$ gates. We can implement it by repetitively entangle the ancilla qubit with its first neighbour and then swapping them. The SNOT gate we created in the previous section does exactly that. In addition to that, Hadamard gates appearing in the circuit are automatically incorporated into our framework thanks to the decoupling gate (see Eq. (5.20)).




(5.39)

All in all, we can easily estimate the resources needed by a direct count of the gates that we need to apply. We can calculate directly the number of atoms necessary.

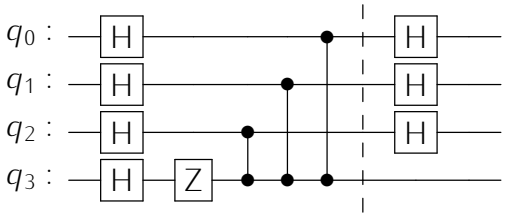



**Figure 5.3:** Deutsch–Jozsa quantum circuit

As it is clear from Fig. (add fig) there is need of the:

- decoupling gate per qubit: $6N$ atoms
- S-NOT per computational qubit: $24(N-1)$
- measurement gate $M$: an additional $6(N-1)$

for a total of $36N-30$. For this reason we can confidently say that the algorithm scales linearly with the number of qubits $N$. The other resources needed can be estimated in a similar fashion and are synthesized in table 5.3.

| Algo. | Atoms | Meas. no. | Change of basis Pulses no. | Driving pulses no. |
|---|---|---|---|---|
| DJ | $36N-10$ | $18N-16$ | $20N-16$ | $33N-29$ |

**Table 5.3:** Resources needed to run Deutsch–Jozsa algorithm.

# Conclusions

In this second part of the thesis we explored Rydberg atoms quantum computation and optimization with a Bayesian approach. We started with a full chapter (Chapter 3) on an optimization method: Bayesian optimization. This was done because variational quantum algorithms have been the protagonists of the NISQ era but there is often not enough focus on the classical part of optimization. In particular it is hard to find an optimizer that can be ran only one time and produce concrete results, this is due to the many source noises on quantum computers and the presence of barren plateaus. For these reason we first introduce Bayesian Optimization with a series of theoretical (simulated) results on digital quantum computers that would show that it can converge to the solution with: few calls to the quantum circuit, few shots to evaluate the energy at every step and in the presence of noise as well. These characteristics are essential for an optimizer to be ran on a computer with a time budget as small as the one we had. The optimization problem we tackled is a classical combinatorial problem on graph, the Max Independent Set (MIS) which is perfect to exploit, in particular, the properties of Rydberg atoms' QPUs.

We followed with Chapter 4, where we introduced the neutral atom platform and its functioning, mainly referring to the structure of the quantum hardware owned by the company PASQAL. In this chapter we showed the results of the experiment run on PASQAL's QPU. With our limited time budget we were able only to run a few sequences as initial benchmark and then run only 4 optimization loops once each. For this reason we had to make very educated decisions. From the results on the initial sequence we realised we could discard 20% of the measured bitstrings in order to fight the measurement error present on the machine at the time, one of the main sources of noise on the machine at the time of writing. This allowed us to gain better results on the following loops. The results from the first three closed optimization showed us we could find the solution to the MIS problem with good agreement and only needing as few as 32 shots per step on a $N = 6$ qubits graph and 64 shots on a $N = 11$ qubits graph.

For the final loop we decided to use all the remaining shots and thus perform a full optimization of 100 steps and 450 shots per step on a $N = 15$ qubits graph. This resulted in a perfect optimization almost in which the energy to minimize decreases almost monotonically, a type of result that is difficult to obtain when considering noise on such a large graph. The optimization was successful since we were able to measure the solution with a good probability and it showed that Bayesian Optimization does maintain the quality during the minimization that we had showed in the previous Chapter.

After showing our results of optimization in the context of both standard digital and analog quantum computation we explored a new proposal for measurement-based quantum computation on neutral atoms that was developed by [Cré22] and we extended adding detailed contributions. MBQC is an alternative and yet almost opposite way to perform computation because it consists in starting from an entangle state and disentangle it by local projective measurements. After introducing this idea we showed another approach that is based on the tensor network representation of the states and allows us to perform the computation on many types of states of interest also in many-body theory. The one selected by [Cré22] was the dimer state created on Rydberg atoms by [SLK+21] and we showed how universal MBQC can be implemented on such a state and some limitations like the fact that the rotation gate had a success rate of 50%. We then added our contribution: we fixed the stochastic problem of the rotation gate, invented a series of other possible gates, rewrote the whole scheme in terms of macro-gate that encompass many measurements and single qubit gates presenting the rules of this new language. This implementation of MBQC implies non-destructive measurements, so it would not be suitable for the current state of the technology we used for the experiment in chapter 4. However, it is already available in other platforms developed for example by the Harvard group [SLK+21].

This last point is exactly the one that might host some additional developments. The fact that it is possible to perform quantum computation in this MBQC framework on the dimer state suggests there might be some theoretical results that show why this state allowed universal quantum computation. Another relevant topic is the connection between the topological properties of the dimer state and the ability to perform quantum computation. Although the theory connecting universal MBQC and Symmetry Protected Topological States (SPTs) is solid showing strong results in $1D$ [SWP+17] and open to interesting extensions in $2D$ [ROW+19], the connection between intrinsic topological states like the dimer state and MBQC is completely unexplored. Finally, the connection with Quantum Error Connection is also very solid and as relevant as ever since we are, at the time of writing, said to be exiting the NISQ era and entering the quantum corrected era. We can say with confidence that whereas MBQC is not the main reference framework for today's Rydberg platforms it is still interesting to consider for all the connections to quantum information and many-body systems and also because as we showed in Sec. 5.3.3 the resources needed scale in favour with the number of qubits.

# Bibliography


[ABKR22] Daniel P. Arovas, Erez Berg, Steven A. Kivelson, and Srinivas Raghu. The hubbard model. *Annual Review of Condensed Matter Physics*, 13(1):239–274, 2022.

[ACC+21] Andrew Arrasmith, M. Cerezo, Piotr Czarnik, Lukasz Cincio, and Patrick J. Coles. Effect of barren plateaus on gradient-free optimization. *Quantum*, 5:558, 2021.

[ADL+23] Boris Albrecht, Constantin Dalyac, Lucas Leclerc, Luis Ortiz-Gutiérrez, Slimane Thabet, Mauro D'Arcangelo, Julia R. K. Cline, Vincent E. Elfving, Lucas Lassablière, Henrique Silvério, Bruno Ximenez, Louis-Paul Henry, Adrien Signoles, and Loïc Henriet. Quantum feature maps for graph machine learning on a neutral atom quantum processor. *Phys. Rev. A*, 107:042615, 2023.

[AKLT87] Ian Affleck, Tom Kennedy, Elliott H. Lieb, and Hal Tasaki. Rigorous results on valence-bond ground states in antiferromagnets. *Phys. Rev. Lett.*, 59:799–802, 1987.

[AOP15] J. Asbóth, László Oroszlány, and András Pályi. *A Short Course on Topological Insulators: Band-structure topology and edge states in one and two dimensions*, volume 919 of *Lecture Notes in Physics*. Springer Nature, 09 2015.

[AV06] David Arthur and Sergei Vassilvitskii. How slow is the k-means method? In *Proceedings of the Twenty-Second Annual Symposium on Computational Geometry*, SCG '06, page 144–153, New York, NY, USA, 2006. Association for Computing Machinery.

[BCJ+00] H.-J. Briegel, T. Calarco, D. Jaksch, J. I. Cirac, and P. Zoller. Quantum computing with neutral atoms. *Journal of Modern Optics*, 47(2-3):415–451, 2000.

[BCMT17] Peter Broecker, Juan Carrasquilla, Roger G. Melko, and Simon Trebst. Machine learning quantum phases of matter beyond the fermion sign problem. *Scientific Reports*, 7(1), 2017.

[BdV08] M Blaauboer and R L de Visser. An analytical decomposition protocol for optimal implementation of two-qubit entangling gates. *Journal of Physics A: Mathematical and Theoretical*, 41(39):395307, 2008.

[BEG+23] Dolev Bluvstein, Simon J. Evered, Alexandra A. Geim, Sophie H. Li, Hengyun Zhou, Tom Manovitz, Sepehr Ebadi, Madelyn Cain, Marcin Kalinowski, Dominik Hangleiter, J. Pablo Bonilla Ataides, Nishad Maskara, Iris Cong, Xun Gao, Pedro Sales Rodriguez, Thomas Karolyshyn, Giulia Semeghini, Michael J. Gullans, Markus Greiner, Vladan Vuletić, and Mikhail D. Lukin. Logical quantum processor based on reconfigurable atom arrays. *Nature*, 2023.

[Ben12] Yoshua Bengio. Deep learning of representations for unsupervised and transfer learning. In Isabelle Guyon, Gideon Dror, Vincent Lemaire, Graham Taylor, and Daniel Silver, editors, *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, volume 27 of *Proceedings of Machine Learning Research*, pages 17–36, Bellevue, Washington, USA, 02 Jul 2012. PMLR.

[Ber13] B. Andrei Bernevig. *Topological Insulators and Topological Superconductors*. Princeton University Press, 2013.

[BH13] B. Andrei Bernevig and Taylor L. Hughes. *Topological Insulators and Topological Superconductors*. Princeton University Press, April 2013.

[BLdL+18] Daniel Barredo, Vincent Lienhard, Sylvain de Léséleuc, Thierry Lahaye, and Antoine Browaeys. Synthetic three-dimensional atomic structures assembled atom by atom. *Nature*, 561(7721):79–82, 2018.

[BLS+22] Dolev Bluvstein, Harry Levine, Giulia Semeghini, Tout T. Wang, Sepehr Ebadi, Marcin Kalinowski, Alexander Keesling, Nishad Maskara, Hannes Pichler, Markus Greiner, Vladan Vuletić, and Mikhail D. Lukin. A quantum processor based on coherent transport of entangled atom arrays. *Nature*, 604(7906):451–456, 2022.

[BR01] Hans J. Briegel and Robert Raussendorf. Persistent entanglement in arrays of interacting particles. *Phys. Rev. Lett.*, 86:910–913, 2001.

[CBB+23] Cheng Chen, Guillaume Bornet, Marcus Bintz, Gabriel Emperauger, Lucas Leclerc, Vincent S. Liu, Pascal Scholl, Daniel Barredo, Johannes Hauschild, Shubhayu Chatterjee, Michael Schuler, Andreas M. Läuchli, Michael P. Zaletel, Thierry Lahaye, Norman Y. Yao, and Antoine Browaeys. Continuous symmetry breaking in a two-dimensional rydberg array. *Nature*, 616(7958):691–695, 2023.

[CCC+19] Giuseppe Carleo, Ignacio Cirac, Kyle Cranmer, Laurent Daudet, Maria Schuld, Naftali Tishby, Leslie Vogt-Maranto, and Lenka Zdeborová . Machine learning and the physical sciences. *Reviews of Modern Physics*, 91(4), 12 2019.

[CD07] Bob Coecke and Ross Duncan. A graphical calculus for quantum observables. *Preprint*, 2007.

[CGG18] Stefanie Czischek, Martin Gärttner, and Thomas Gasenzer. Quenches near ising quantum criticality as a challenge for artificial neural networks. *Phys. Rev. B*, 98:024311, 2018.

[CGLN20] Yanming Che, Clemens Gneiting, Tao Liu, and Franco Nori. Topological quantum phase transitions retrieved through unsupervised machine learning. *Phys. Rev. B*, 102:134213, 2020.

[CHMT23] Ming-Chiang Chung, Guang-Yu Huang, Ian P. McCulloch, and Yuan-Hong Tsai. Deep learning of phase transitions for quantum spin chains from correlation aspects. *Phys. Rev. B*, 107:214451, 2023.

[CLN05] Andrew M. Childs, Debbie W. Leung, and Michael A. Nielsen. Unified derivations of measurement-based schemes for quantum computation. *Physical Review A*, 71(3), 2005.

[CM17] Juan Carrasquilla and Roger G. Melko. Machine learning phases of matter. *Nature Physics*, 13, 2017.

[CNL+16] Lawrence W. Cheuk, Matthew A. Nichols, Katherine R. Lawrence, Melih Okan, Hao Zhang, Ehsan Khatami, Nandini Trivedi, Thereza Paiva, Marcos Rigol, and Martin W. Zwierlein. Observation of spatial charge and spin correlations in the 2d fermi-hubbard model. *Science*, 353(6305):1260–1264, 2016.

[CPGSV21] J. Ignacio Cirac, David Pérez-García, Norbert Schuch, and Frank Verstraete. Matrix product states and projected entangled pair states: Concepts, symmetries, theorems. *Reviews of Modern Physics*, 93(4), 2021.

[Cré22] Valentin Crépel. Dimer states of rydberg atoms on the kagome lattice as resources for universal measurement-based quantum computation. *AIP Advances*, 12(11):115301, 2022.

[CSV+21] 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):1791, 2021.

[CT17] Giuseppe Carleo and Matthias Troyer. Solving the quantum many-body problem with artificial neural networks. *Science*, 355(6325):602–606, February 2017.

[CTEP24] Filippo Caleca, Simone Tibaldi, Elisa Ercolessi, and Guido Pupillo. Unsupervised machine learning study of an extended hubbard model. *In preparation*, 2024.

[CTSR16] Ching-Kai Chiu, Jeffrey C. Y. Teo, Andreas P. Schnyder, and Shinsei Ryu. Classification of topological quantum matter with symmetries. *Rev. Mod. Phys.*, 88:035005, 2016.

[DAR+22] Anna Dawid, Julian Arnold, Borja Requena, Alexander Gresch, Marcin Płodzień, Kaelan Donatella, Kim A. Nicoli, Paolo Stornati, Rouven Koch, Miriam Büttner, Robert Okuła, Gorka Muñoz-Gil, Rodrigo A. Vargas-Hernández, Alba Cervera-Lierta, Juan Carrasquilla, Vedran Dunjko, Marylou Gabrié, Patrick Huembeli, Evert van Nieuwenburg, Filippo Vicentini, Lei Wang, Sebastian J. Wetzel, Giuseppe Carleo, Eliška Greplová, Roman Krems, Florian Marquardt, Michał Tomza,

Maciej Lewenstein, and Alexandre Dauphin. Modern applications of machine learning in quantum sciences, 2022.

[DHK+23] Constantin Dalyac, Louis-Paul Henry, Minhyuk Kim, Jaewook Ahn, and Loïc Henriet. Exploring the impact of graph locality for the resolution of the maximum- independent-set problem with neutral atom devices. *Phys. Rev. A*, 108:052423, 2023.

[DHTPac21] Jean-Yves Desaules, Ana Hudomal, Christopher J. Turner, and Zlatko Papić. Proposal for realizing quantum scars in the tilted 1d fermi-hubbard model. *Phys. Rev. Lett.*, 126:210601, 2021.

[DKP07] Vincent Danos, Elham Kashefi, and Prakash Panangaden. The measurement calculus. *Journal of the ACM*, 54(2):8, 2007.

[DLC+15] M. Dalmonte, W. Lechner, Zi Cai, M. Mattioli, A. M. Läuchli, and G. Pupillo. Cluster luttinger liquids and emergent supersymmetric conformal critical points in the one-dimensional soft-shoulder hubbard model. *Physical Review B*, 92(4), 2015.

[DP99] Ding-Zhu Du and Panos M. Pardalos, editors. *Handbook of Combinatorial Optimization*. Springer, 1999.

[DRTG22] Xinyang Dong, Lorenzo Del Re, Alessandro Toschi, and Emanuel Gull. Mechanism of superconductivity in the hubbard model at intermediate interaction strength. *Proceedings of the National Academy of Sciences*, 119(33), 2022.

[Edw73] C. S. Edwards. Some extremal properties of bipartite subgraphs. *Canadian Journal of Mathematics*, 25(3):475–485, 1973.

[EKC+22] S. Ebadi, A. Keesling, M. Cain, T. T. Wang, H. Levine, D. Bluvstein, G. Semeghini, A. Omran, J.-G. Liu, R. Samajdar, X.-Z. Luo, B. Nash, X. Gao, B. Barak, E. Farhi, S. Sachdev, N. Gemelke, L. Zhou, S. Choi, H. Pichler, S.-T. Wang, M. Greiner, V. Vuletić, and M. D. Lukin. Quantum optimization of maximum independent set using rydberg atom arrays. *Science*, 376(6598):1209–1215, 2022.

[EWL+21] Sepehr Ebadi, Tout T. Wang, Harry Levine, Alexander Keesling, Giulia Semeghini, Ahmed Omran, Dolev Bluvstein, Rhine Samajdar, Hannes Pichler, Wen Wei Ho, Soonwon Choi, Subir Sachdev, Markus Greiner, Vladan Vuletić, and Mikhail D. Lukin. Quantum phases of matter on a 256-atom programmable quantum simulator. *Nature*, 595(7866):227–232, 2021.

[FGG14] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm, 2014.

[FMV+20] P. Fromholz, G. Magnifico, V. Vitale, T. Mendes-Santos, and M. Dalmonte. Entanglement topological invariants for one-dimensional topological superconductors. *Phys. Rev. B*, 101:085136, 2020.

[Fra18] Peter I. Frazier. A tutorial on Bayesian optimization, 2018.

[FWS20] Matthew Fishman, Steven R. White, and E. Miles Stoudenmire. The ITensor software library for tensor network calculations, 2020.

[GE07] D. Gross and J. Eisert. Novel schemes for measurement-based quantum computation. *Phys. Rev. Lett.*, 98:220503, 2007.

[GE21] Marek Gluza and Jens Eisert. Recovering quantum correlations in optical lattices from interaction quenches. *Phys. Rev. Lett.*, 127:090503, 2021.

[GESPG07] D. Gross, J. Eisert, N. Schuch, and D. Perez-Garcia. Measurement-based quantum computation beyond the one-way model. *Phys. Rev. A*, 76:052315, 2007.

[Gia04] Thierry Giamarchi. *Quantum Physics in One Dimension*. International Series of Monographs on Physics. Clarendon Press, 2004.

[GJ78] M. R. Garey and D. S. Johnson. " strong " np-completeness results: Motivation, examples, and implications. *J. ACM*, 25(3):499–508, 1978.

[GJ82] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Co, 1982.

[GS17] Gian Giacomo Guerreschi and Mikhail Smelyanskiy. Practical optimization for hybrid quantum-classical algorithms, 2017.

[GVB+20] Eliska Greplova, Agnes Valenti, Gregor Boschung, Frank Schäfer, Niels Lörch, and Sebastian D Huber. Unsupervised identification of topological phase transitions using predictive models. *New Journal of Physics*, 22(4):045003, 2020.

[HBS+20] Loïc Henriet, Lucas Beguin, Adrien Signoles, Thierry Lahaye, Antoine Browaeys, Georges-Olivier Reymond, and Christophe Jurczak. Quantum computing with neutral atoms. *Quantum*, 4:327, 2020.

[HDW18] Patrick Huembeli, Alexandre Dauphin, and Peter Wittek. Identifying quantum phase transitions with adversarial neural networks. *Physical Review B*, 97(13), 2018.

[HFJea17] T. Hensgens, T. Fujita, L. Janssen, and et al. Quantum simulation of a fermi–hubbard model using a semiconductor quantum dot array. *Nature*, 548:70–73, 2017.

[HKP20] 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.

[HS12] Fabian Hassler and Dirk Schuricht. Strongly interacting Majorana modes in an array of josephson junctions. *New Journal of Physics*, 14(12):125018, 2012.

[HSN+21] Matthew P. Harrigan, Kevin J. Sung, Matthew Neeley, Kevin J. Satzinger, Frank Arute, Kunal Arya, Juan Atalaya, Joseph C. Bardin,

Rami Barends, Sergio Boixo, and et al. Quantum approximate optimization of non-planar graph problems on a planar superconducting processor. *Nature Physics*, 17(3):332–336, 2021.

[HSS17] Wenjian Hu, Rajiv R. P. Singh, and Richard T. Scalettar. Discovering phases, phase transitions, and crossovers through unsupervised machine learning: A critical examination. *Phys. Rev. E*, 95:062122, 2017.

[HTDH21] Louis-Paul Henry, Slimane Thabet, Constantin Dalyac, and Loïc Henriet. Quantum evolution kernel: Machine learning on graphs with programmable arrays of qubits. *Phys. Rev. A*, 104:032416, 2021.

[JNN13] J.R. Johansson, P.D. Nation, and Franco Nori. Qutip 2: A python framework for the dynamics of open quantum systems. *Computer Physics Communications*, 184(4):1234–1240, 2013.

[KdBvW+17] Jorrit Kruthoff, Jan de Boer, Jasper van Wezel, Charles L. Kane, and Robert-Jan Slager. Topological classification of crystalline insulators through band structure combinatorics. *Phys. Rev. X*, 7:041069, 2017.

[KDK+21] Niklas Käming, Anna Dawid, Korbinian Kottmann, Maciej Lewenstein, Klaus Sengstock, Alexandre Dauphin, and Christof Weitenberg. Unsupervised machine learning of topological phase transitions from experimental data. *Machine Learning: Science and Technology*, 2(3):035037, 2021.

[Kit01] A Yu Kitaev. Unpaired majorana fermions in quantum wires. *Physics-Uspekhi*, 44(10S):131–136, October 2001.

[Kit03] A.Yu. Kitaev. Fault-tolerant quantum computation by anyons. *Annals of Physics*, 303(1):2–30, 2003.

[KST15] Hosho Katsura, Dirk Schuricht, and Masahiro Takahashi. Exact ground states and topological order in interacting Kitaev/Majorana chains. *Phys. Rev. B*, 92:115137, 2015.

[KW20] Aleks Kissinger and John Wetering. Pyzx: Large scale automated diagrammatic reasoning. *Electronic Proceedings in Theoretical Computer Science*, 318:230–242, 04 2020.

[LG08] C. Lobo and S. D. Gensemer. Technique for measuring correlation functions in interacting gases. *Phys. Rev. A*, 78:023618, 2008.

[LN89] Dong C. Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical Programming*, 45(1):503–528, 1989.

[LP17] Ville Lahtinen and Jiannis K. Pachos. A Short Introduction to Topological Quantum Computation. *SciPost Phys.*, 3:021, 2017.

[Luc14] Andrew Lucas. Ising formulations of many NP problems. *Frontiers in Physics*, 2, 2014.

[Lut04] J. M. Luttinger. An Exactly Soluble Model of a Many-Fermion System. *Journal of Mathematical Physics*, 4(9):1154–1162, 12 2004.

[LYTS20] Eran Lustig, Or Yair, Ronen Talmon, and Mordechai Segev. Identifying topological phase transitions in experiments using manifold learning. *Phys. Rev. Lett.*, 125:127401, 2020.

[Mac03] David J. C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Copyright Cambridge University Press, 2003.

[Mal13] Mariana Malard. Sine-gordon model: Renormalization group solution and applications. *Brazilian Journal of Physics*, 43(3):182–198, 2013.

[MBB+18] 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*, 3(3):030503, 2018.

[MBS+18] 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):4812, 2018.

[MBW+19] Pankaj Mehta, Marin Bukov, Ching-Hao Wang, Alexandre G.R. Day, Clint Richardson, Charles K. Fisher, and David J. Schwab. A high-bias, low-variance introduction to machine learning for physicists. *Physics Reports*, 810:1–124, 2019. A high-bias, low-variance introduction to Machine Learning for physicists.

[MC21] Matija Medvidović and Giuseppe Carleo. Classical variational simulation of the quantum approximate optimization algorithm. *npj Quantum Information*, 7(1):101, 2021.

[MDLP13] Marco Mattioli, Marcello Dalmonte, Wolfgang Lechner, and Guido Pupillo. Cluster luttinger liquids of rydberg-dressed atoms in optical lattices. *Physical Review Letters*, 111(16), 10 2013.

[MFS19] Glen Bigan Mbeng, Rosario Fazio, and Giuseppe E. Santoro. Quantum annealing: a journey through digitalization, control, and hybrid quantum variational schemes. *arXiv: Quantum Physics*, 2019.

[MGW+03] Olaf Mandel, Markus Greiner, Artur Widera, Tim Rom, Theodor W. Hänsch, and Immanuel Bloch. Controlled collisions for multi-particle entanglement of optically trapped atoms. *Nature*, 425(6961):937–940, 2003.

[Mia17] Jian-Jian Miao. Exact solution for the interacting kitaev chain at the symmetric point. *Physical Review Letters*, 118(26), 2017.

[MLIdJ22] Juliane Mueller, Wim Lavrijsen, Costin Iancu, and Wibe de Jong. Accelerating Noisy VQE Optimization with Gaussian Processes, 4 2022.

[MMS+22] Antonio Anna Mele, Glen Bigan Mbeng, Giuseppe Ernesto Santoro, Mario Collura, and Pietro Torta. Avoiding barren plateaus via transferability of smooth solutions in Hamiltonian Variational Ansatz. *arXiv: Quantum Physics*, 6 2022.

[MRBAG16] Jarrod R. McClean, Jonathan Romero, Ryan Babbush, and Alán Aspuru-Guzik. The theory of variational hybrid quantum-classical algorithms. *New Journal of Physics*, 18(2):023023, 2016.

[MSP02] G. Misguich, D. Serban, and V. Pasquier. Quantum dimer model on the kagome lattice: Solvable dimer-liquid and ising gauge theory. *Phys. Rev. Lett.*, 89:137202, 2002.

[MZvN+21] Paolo Molignini, Antonio Zegarra, Evert van Nieuwenburg, R. Chitra, , and Wei Chen. A supervised learning algorithm for interacting topological insulators based on local curvature. *SciPost Phys.*, 11:73, 2021.

[Nak00] Masaaki Nakamura. Tricritical behavior in the extended hubbard chains. *Phys. Rev. B*, 61:16377–16392, 2000.

[NCH+12] Yuezhen Niu, Suk Bum Chung, Chen-Hsuan Hsu, Ipsita Mandal, S. Raghu, and Sudip Chakravarty. Majorana zero modes in a quantum ising chain with longer-ranged interactions. *Phys. Rev. B*, 85:035110, 2012.

[NEM+23] Piero Naldesi, Andreas Elben, Anna Minguzzi, David Clément, Peter Zoller, and Benoît Vermersch. Fermionic correlation functions from randomized measurements in programmable atomic quantum devices. *Phys. Rev. Lett.*, 131:060601, 2023.

[Nie03] Michael A. Nielsen. Quantum computation by measurement and quantum memory. *Physics Letters A*, 308:96–100, 2003.

[O+17] J. S. Otterbach et al. Unsupervised machine learning on a hybrid quantum computer, 2017.

[PBB+20] Guido Pagano, Aniruddha Bapat, Patrick Becker, Katherine S. Collins, Arinjoy De, Paul W. Hess, Harvey B. Kaplan, Antonis Kyprianidis, Wen Lin Tan, Christopher Baldwin, Lucas T. Brady, Abhinav Deshpande, Fangli Liu, Stephen Jordan, Alexey V. Gorshkov, and Christopher Monroe. Quantum approximate optimization of the long-range ising model with a trapped-ion quantum simulator. *Proceedings of the National Academy of Sciences*, 117(41):25396–25401, 2020.

[PMC+16] Maxwell F. Parsons, Anton Mazurenko, Christie S. Chiu, Geoffrey Ji, Daniel Greif, and Markus Greiner. Site-resolved measurement of the spin-correlation function in the fermi-hubbard model. *Science*, 353(6305):1253–1256, 2016.

[PMS+14] Alberto Peruzzo, Jarrod R. 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):4213, 2014.

[Pre18] John Preskill. Quantum computing in the NISQ era and beyond. *Quantum*, 2:79, 2018.

[PSL06] Kenneth Price, Rainer M Storn, and Jouni A Lampinen. *Differential evolution: a practical approach to global optimization*. Springer Science & Business Media, 2006.

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[PW09] David Poulin and Pawel Wocjan. Preparing ground states of quantum many-body systems on a quantum computer. *Physical Review Letters*, 102(13), 2009.

[RB99] Stanisław Robaszkiewicz and Bogdan R. Bułka. Superconductivity in the hubbard model with pair hopping. *Physical Review B*, 59(9):6430–6437, 1999.

[RB01a] Robert Raussendorf and Hans J. Briegel. Computational model underlying the one-way quantum computer. *Quantum Inf. Comput.*, 2:443–486, 2001.

[RB01b] Robert Raussendorf and Hans J. Briegel. A one-way quantum computer. *Phys. Rev. Lett.*, 86:5188–5191, 2001.

[RBB01] Robert Raussendorf, Dan Browne, and Hans Briegel. The one-way quantum computer - a non-network model of quantum computation. *Journal of Modern Optics*, 49, 09 2001.

[RHG06] R. Raussendorf, J. Harrington, and K. Goyal. A fault-tolerant one-way quantum computer. *Annals of Physics*, 321(9):2242–2270, 2006.

[RLKM22] Monika Richter-Laskowska, Marcin Kurpas, and Maciej Maśka. A learning by confusion approach to characterize phase transitions, 2022.

[RNS19] Joaquin F. Rodriguez-Nieva and Mathias S. Scheurer. Identifying topological order through unsupervised machine learning. *Nature Physics*, 15(8):790–795, 2019.

[ROW+19] Robert Raussendorf, Cihan Okay, Dong-Sheng Wang, David T. Stephen, and Hendrik Poulsen Nautrup. Computationally universal phase of quantum matter. *Physical Review Letters*, 122(9), 2019.

[RW05] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

[Sac11] Subir Sachdev. *Quantum Phase Transitions*. Cambridge University Press, Cambridge, England, second edition, 2011.

[SASF11] E. M. Stoudenmire, Jason Alicea, Oleg A. Starykh, and Matthew P.A. Fisher. Interaction effects in topological superconducting wires supporting Majorana fermions. *Phys. Rev. B*, 84:014503, 2011.

[Sch05] U. Schollwöck. The density-matrix renormalization group. *Rev. Mod. Phys.*, 77:259–315, 2005.

[Sch11] Ulrich Schollwöck. The density-matrix renormalization group in the age of matrix product states. *Annals of Physics*, 326(1):96–192, 1 2011.

[SGD+22] Henrique Silvério, Sebastián Grijalva, Constantin Dalyac, Lucas Leclerc, Peter J. Karalekas, Nathan Shammah, Mourad Beji, Louis-Paul Henry, and Loïc Henriet. Pulser: An open-source package for the design of pulse sequences in programmable neutral-atom arrays. *Quantum*, 6:629, 2022.

[SKS+21] Chris N. Self, Kiran E. Khosla, Alistair W. R. Smith, Frédéric Sauvage, Peter D. Haynes, Johannes Knolle, Florian Mintert, and M. S. Kim. Variational quantum algorithm with information sharing. *npj Quantum Information*, 7(1):116, 2021.

[SKS23] Ryan Shaffer, Lucas Kocia, and Mohan Sarovar. Surrogate-based optimization for variational quantum algorithms. *Phys. Rev. A*, 107:032415, 2023.

[SLA12] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical Bayesian optimization of machine learning algorithms. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'12, pages 2951–2959, Red Hook, NY, USA, 2012. Curran Associates Inc.

[SLK+21] G. Semeghini, H. Levine, A. Keesling, S. Ebadi, T. T. Wang, D. Bluvstein, R. Verresen, H. Pichler, M. Kalinowski, R. Samajdar, A. Omran, S. Sachdev, A. Vishwanath, M. Greiner, V. Vuletić, and M. D. Lukin. Probing topological spin liquids on a programmable quantum simulator. *Science*, 374(6572):1242–1247, 2021.

[SMJZ13] Robert-Jan Slager, Andrej Mesaros, Vladimir Juričić, and Jan Zaanen. The space group classification of topological band-insulators. *Nature Physics*, 9(2):98–102, 2013.

[SRFL08] Andreas P. Schnyder, Shinsei Ryu, Akira Furusaki, and Andreas W. W. Ludwig. Classification of topological insulators and superconductors in three spatial dimensions. *Phys. Rev. B*, 78:195125, 2008.

[SS20] Mathias S. Scheurer and Robert-Jan Slager. Unsupervised machine learning and band topology. *Phys. Rev. Lett.*, 124:226401, 2020.

[SSW+16] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

[SWM10] M. Saffman, T. G. Walker, and K. Mølmer. Quantum information with Rydberg atoms. *Rev. Mod. Phys.*, 82:2313–2363, 2010.

[SWP+17] David T. Stephen, Dong-Sheng Wang, Abhishodh Prakash, Tzu-Chieh Wei, and Robert Raussendorf. Computational power of symmetry-protected topological phases. *Physical Review Letters*, 119(1), 2017.

[SYZ+18] Ning Sun, Jinmin Yi, Pengfei Zhang, Huitao Shen, and Hui Zhai. Deep learning topological invariants of band insulators. *Physical Review B*, 98(8), 2018.

[Tab11] Noel M. Tabia. Quantum computing with cluster states gelo. In *Quantum Computing with Cluster States Gelo*, 2011.

[Tas97] Hal Tasaki. The hubbard model: Introduction and selected rigorous results, 1997.

[TMC+18] Giacomo Torlai, Guglielmo Mazzola, Juan Carrasquilla, Matthias Troyer, Roger Melko, and Giuseppe Carleo. Neural-network quantum state tomography. *Nature Physics*, 14(5):447–450, 2018.

[TMVE23] Simone Tibaldi, Giuseppe Magnifico, Davide Vodola, and Elisa Ercolessi. Unsupervised and supervised learning of interacting topological phases from single-particle correlation functions. *SciPost Phys.*, 14:005, 2023.

[Tom50] Sin-itiro Tomonaga. Remarks on Bloch's Method of Sound Waves applied to Many-Fermion Problems. *Progress of Theoretical Physics*, 5(4):544–569, 07 1950.

[TRS13] Ronny Thomale, Stephan Rachel, and Peter Schmitteckert. Tunneling spectra simulation of interacting Majorana wires. *Phys. Rev. B*, 88:161103, 2013.

[TSP18] Leticia Tarruell and Laurent Sanchez-Palencia. Quantum simulation of the hubbard model with ultracold fermions in optical lattices. *Comptes Rendus Physique*, 19(6):365–393, 2018.

[TVTE23] Simone Tibaldi, Davide Vodola, Edoardo Tignone, and Elisa Ercolessi. Bayesian optimization for qaoa. *IEEE Transactions on Quantum Engineering*, 4:1–11, 2023.

[TY22] Shiro Tamiya and Hayata Yamasaki. Stochastic gradient line Bayesian optimization for efficient noise-robust optimization of parameterized quantum circuits. *npj Quantum Information*, 8(1):90, 2022.

[VC04] Frank Verstraete and J. I. Cirac. Valence-bond states for quantum computation. *Physical Review A*, 70(6):060302–, 2004.

[vdMH08] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 11 2008.

[VGO+20] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson,

Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[VLE+14] Davide Vodola, Luca Lepori, Elisa Ercolessi, Alexey V. Gorshkov, and Guido Pupillo. Kitaev chains with long-range pairing. *Phys. Rev. Lett.*, 113:156402, 2014.

[VLV21] Ruben Verresen, Mikhail D. Lukin, and Ashvin Vishwanath. Prediction of toric code topological order from rydberg blockade. *Physical Review X*, 11(3), 2021.

[vNLH17a] Evert P. L. van Nieuwenburg, Ye-Hua Liu, and Sebastian D. Huber. Learning phase transitions by confusion. *Nature Physics*, 13(5):435–439, February 2017.

[vNLH17b] Evert P. L. van Nieuwenburg, Ye-Hua Liu, and Sebastian D. Huber. Learning phase transitions by confusion. *Nature Physics*, 13(5):435–439, 2017.

[Wan16] Lei Wang. Discovering phase transitions with unsupervised learning. *Phys. Rev. B*, 94:195105, 2016.

[WD97] David J. Wales and Jonathan P. K. Doye. Global optimization by basin-hopping and the lowest energy structures of lennard-jones clusters containing up to 110 atoms. *The Journal of Physical Chemistry A*, 101(28):5111–5116, 1997.

[Wei21] Tzu-Chieh Wei. Measurement-based quantum computation. *Oxford Research Encyclopedia of Physics*, 2021.

[WHJR18] Zhihui Wang, Stuart Hadfield, Zhang Jiang, and Eleanor G. Rieffel. Quantum approximate optimization algorithm for maxcut: A fermionic view. *Phys. Rev. A*, 97:022304, 2018.

[WHT16] Dave Wecker, Matthew B. Hastings, and Matthias Troyer. Training a quantum optimizer. *Phys. Rev. A*, 94:022309, 2016.

[WKJC21] Guoming Wang, Dax Enshan Koh, Peter D. Johnson, and Yudong Cao. Minimizing estimation runtime on noisy quantum computers. *PRX Quantum*, 2:010346, 2021.

[XSFG97] Y Xiang, D.Y Sun, W Fan, and X.G Gong. Generalized simulated annealing algorithm and its application to the thomson model. *Physics Letters A*, 233(3):216–220, 1997.

[YD21] Li-Wei Yu and Dong-Ling Deng. Unsupervised learning of non-hermitian topological phases. *Phys. Rev. Lett.*, 126:240402, 2021.

[YZSD23] Li-Wei Yu, Shun-Yao Zhang, Pei-Xin Shen, and Dong-Ling Deng. Unsupervised learning of interacting topological phases from experimental observables. *Fundamental Research*, 2023.

[ZCZW19] B. Zeng, X. Chen, D-L. Zhou, and X-G Wen. *Quantum Information Meets Quantum Matter*. Springer, 2019.

[ZLB+19] D. Zhu, N. M. Linke, M. Benedetti, K. A. Landsman, N. H. Nguyen, C. H. Alderete, A. Perdomo-Ortiz, N. Korda, A. Garfoot, C. Brecque, L. Egan, O. Perdomo, and C. Monroe. Training of quantum circuits on a hybrid quantum computer. *Science Advances*, 5(10), 2019.

[ZSZ18] Pengfei Zhang, Huitao Shen, and Hui Zhai. Machine learning topological invariants with neural networks. *Physical Review Letters*, 120(6), 2018.

[ZWC+20] 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. *Phys. Rev. X*, 10:021067, 2020.

# A | Details of Bayesian optimization

In this appendix we give a general overview of the Bayesian terms used in Chapter 3 and a detailed review of the algorithm we used in this work to perform the Bayesian optimization for the QAOA.

## A.1 Overview of Bayesian terminology

The Gaussian process is a surrogate model which aims at reconstructing the landscape of optimization of an unknown function $f(\boldsymbol{\theta})$. It is one of the two main ingredients of the Bayesian optimization algorithm, along with the acquisition function. We can now define the Bayesian terms used throughout Chapter 3.

1. Prior $p(\boldsymbol{f})$. This distribution encapsulates the previous knowledge we have about the target function $f$. Typically, it consists in a multivariate normal distribution centered around 0 in which the covariance between points is assigned by a kernel function like Eq.(3.8):

$$p(\boldsymbol{f}) = \mathcal{N}(\mathbf{0}, k(x, x')). \tag{A.1}$$

   To make an initial guess on our function we can sample a function $\boldsymbol{f}^*$ from this distribution. To do so, we choose a set of points $\boldsymbol{\Theta}$ and evaluate:

$$\boldsymbol{f}^* \sim \mathcal{N}(\mathbf{0}, k(\boldsymbol{\Theta}, \boldsymbol{\Theta})). \tag{A.2}$$

2. Likelihood $p(\boldsymbol{y}|\boldsymbol{f})$. This distribution represents the compatibility between the prior $p(\boldsymbol{f})$ and the observations $\boldsymbol{y}$. In the context of Gaussian processes it is defined by another Gaussian distribution.

3. Posterior $p(\boldsymbol{f}|\boldsymbol{y})$. This describes the knowledge we have about $f$ after having collected some observations $\boldsymbol{y} = f(\boldsymbol{\Theta})$. The aim of the Gaussian process is to make the posterior generate functions as similar as possible to $f$. It is related to the prior and likelihood through the Bayesian theorem:

$$p(\boldsymbol{f}|\boldsymbol{y}) = \frac{p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f})}{p(\boldsymbol{y})}, \tag{A.3}$$

   which states that a posterior distribution is proportional to their product. In the context of Gaussian processes, the posterior is calculated from the prior by an operation which is called *conditioning*, resulting in

$$p(\boldsymbol{f}|\boldsymbol{y}) \sim \mathcal{N}(\mu', k'). \tag{A.4}$$

Where the new mean and covariance are defined in the text (Eqs. (3.10), (3.11)). From Eq. (A.4) we see that the posterior is itself a Gaussian multivariate distribution. Therefore, we can sample functions $\boldsymbol{f}^*$ as we do with the prior (A.2) but now their values will coincide with $\boldsymbol{y}$ at every point $\boldsymbol{\Theta}$ where we sampled $f$.

4. Marginal likelihood $p(\boldsymbol{y})$. Also called *evidence*, it is the normalization term in Eq. (A.3). In Bayesian inference it represents the total probability of generating the observed samples $\boldsymbol{f}$ from the prior. It is indeed obtained integrating over all possible function values $\boldsymbol{f}$:

$$p(\boldsymbol{y}) = \int p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f})d\boldsymbol{f}. \tag{A.5}$$

Like the posterior distribution, it has a closed form in term of a multivariate normal distribution. Thus, the maximization of its logarithm is used in Gaussian processes to pick the best hyperparameters (as done in Sec. 3.2.4).

### A.1.1 Review of Bayesian optimization algorithm

This algorithm is made out of three phases: warm-up, kernel optimization, acquisition function maximization and is summarized in Algorithm 1.

*Warm-up* – In the warm-up phase we start with a set of $N_W = 10$ points $X = \{\boldsymbol{\theta}_j\}_{j=1}^{N_W}$ with $\boldsymbol{\theta}_j \in \mathbb{R}^d$ where $d = 2p$ and $p$ is the depth of the QAOA circuit. Each point $\boldsymbol{\theta}_j$ is a set of angles $(\boldsymbol{\gamma}, \boldsymbol{\beta})$. These points are sampled from the latin hypercube of bounds $[0, \pi]^d$. For each set of angles we estimate the energy of QAOA $y_j$ and we create the design matrix $\boldsymbol{\Theta} = (\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_{N_W})$ and the observation vector $\boldsymbol{y}$ with the energies of the points $\boldsymbol{\theta}_j$. We also store the point with lowest energy as $(\boldsymbol{\theta}_m, f_m)$. At this step there is no calculation involving the Gaussian process which is currently set to its prior: a multinomial gaussian distribution centered around zero.

*Kernel optimization* – In this part we look for the hyperparameters $(\tilde{\sigma}, \tilde{\ell})$ which maximize the marginal likelihood function $p(\boldsymbol{y}|\boldsymbol{\Theta})$ (Eq. (3.14)). This optimization is performed by repeating the `L-BFGS` [LN89] minimization on $-p(\boldsymbol{y}|\boldsymbol{\Theta})$ for 10 times and selecting the best parameters found. The parameters found at every step of the optimization are plotted in panels (c) and (d) of Fig. A.2.

*Acquisition function maximization* – Once the hyperparameters are set the algorithm exploits its knowledge and uncertainty of the data to propose a new point $\boldsymbol{\theta}'$ with the new parameters where we evaluate the QAOA circuit. This is done by maximizing the expected improvement in Eq. (3.12).

The new point $\boldsymbol{\theta}'$ maximizing the expected improvement is then added to the dataset $\mathcal{D}$ and the algorithm is repeated. The procedure stops after $N_{\text{BAYES}}$ iterations. Fig. A.1 provides an illustrative example (with $p = 1$, $N_W = 5$, and $N_{\text{BAYES}} = 20$) of how the Bayesian method operates and moves through the landscapes.

## A.2 Optimizers beyond Bayesian Optimization

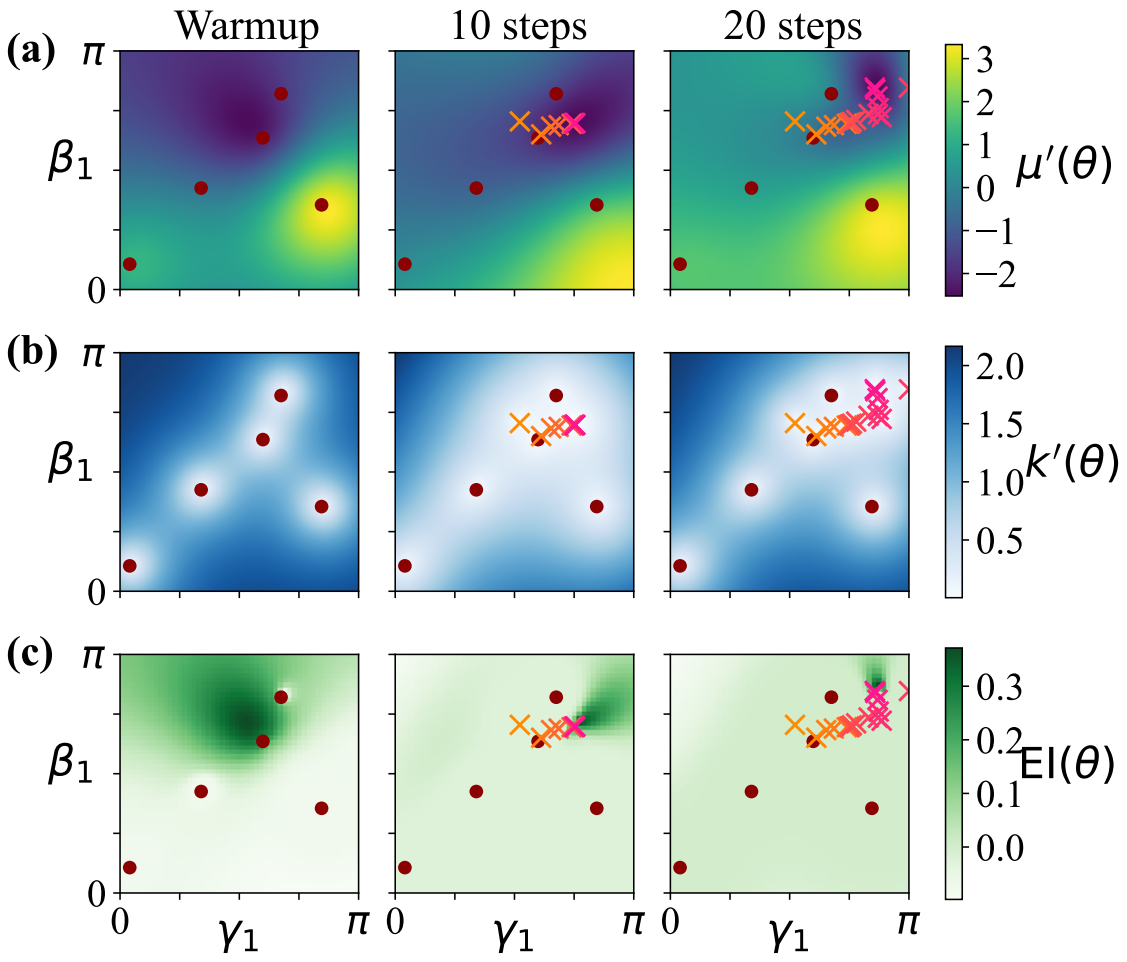



**Figure A.1: Posterior and acquisition function during optimization.** Here we plot the posterior and the acquisition function at three different steps of the optimization: (from left to right) at warm-up, after 10 steps and after 20 steps. In rows (a) and (b) we plot respectively the mean $\mu'(\boldsymbol{\theta})$ and the variance $k'(\boldsymbol{\theta})$ of the posterior while in row (c) the acquisition function EI($\boldsymbol{\theta}$) (see Eqs. (3.10) - (3.12) in the text). These data, obtained running the Bayesian optimization on QAOA with $p = 1$ on the graph of Fig. 3.1(a), are shown as a function of the two variational parameters $\boldsymbol{\theta} = (\gamma_1, \beta_1)$. The red dots indicate the warm-up points ($N_W = 5$, in this case) while the crosses are the points that have been selected by subsequent Bayesian optimization steps as new candidate solutions (with a color scale from first (orange) to last (pink)). At every Bayesian optimization step, $\mu'(\boldsymbol{\theta})$ encodes the knowledge of the landscape of the function to optimize, while $k'(\boldsymbol{\theta})$ contains the uncertainty. The acquisition function EI($\boldsymbol{\theta}$) combines the information from $\mu'$ and $k'$ and the position of its maximum proposes the next possible optimal point. From these considerations, we see that after 20 steps the mean $\mu'(\boldsymbol{\theta})$ (row (a)) recreates the landscape (our knowledge of $E(\boldsymbol{\theta}_1)$) with more precision than at warm-up (compare with the real energy landscape in Fig. 3.2(a)). In addition to that, EI($\boldsymbol{\theta}$) becomes more and more flat in all the landscape except for a small area in the top right corner on the right panel of (c). This means that the Gaussian process has acquired enough knowledge from the data to converge to the energy minimum.

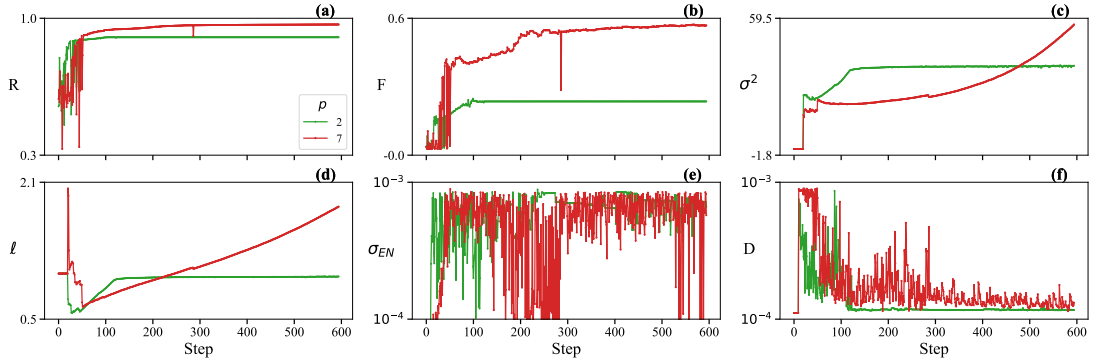



**Figure A.2: Parameters of Bayesian optimization at p = 2, 7.** Plots of the parameters changing during Bayesian optimization for two runs with $N_{BAYES} = 600$ steps. (a) Approximation ratio $R$. (b) Fidelity $F$. (c) Kernel constant $\sigma^2$. (d) Kernel correlation length $\ell$. (e) Standard deviation of the expected improvement and (f) average distance of the points of differential evolution at the last generation $N_T$.

### A.2.1 Differential evolution

Finding the point $\tilde{\boldsymbol{\theta}}$ of the parameter space that maximizes the expected improvement $\mathrm{EI}(\boldsymbol{\theta})$ (Eq. 3.12) is not an easy task since $\mathrm{EI}(\boldsymbol{\theta})$ can show a fairly flat landscape [SLA12], in particular after many optimization steps (for example, see Fig. A.1(c)).

To compute the maximum of $\mathrm{EI}(\boldsymbol{\theta})$ in this work we use the differential evolution algorithm [PSL06]. This is an evolutionary method in which populations of points $\{\boldsymbol{\theta}\}$, called generations, are iteratively obtained from the previous ones until convergence. The algorithm starts by initializing a generation (a) and then the population is updated following three main steps: (b) mutation, (c) cross-over and (d) selection.

(a) We choose as starting population $N_P = 15 \cdot 2p$ points $\{\boldsymbol{\theta}_{i,1}\}$ where the index $i \in \{1, \dots, N_P\}$ uniquely identifies the point within the belonging population, while the index 1 indicates that the point belongs to the first generation $g = 1$. These points are randomly generated on the latin $2p$-cube of bounds $[0, \pi]^{2p}$ and to each point there is an associated expected improvement $\mathrm{EI}(\boldsymbol{\theta}_{i,1})$.

(b) For each $\boldsymbol{\theta}_{i,g}$ (called parent point) in the population, the differential evolution picks three random points, different from $\boldsymbol{\theta}_{i,g}$, labelled by $r_0, r_1, r_2$ within the corresponding population, and creates a new point as:

$$\boldsymbol{v}_{i,g} = \boldsymbol{\theta}_{r_0,g} + M(\boldsymbol{\theta}_{r_1,g} - \boldsymbol{\theta}_{r_2,g}), \tag{A.6}$$

where $M \in (0.5, 1)$ is a hyperparameter that is selected randomly at every generation. Through Eq. A.6 the differential evolution mutates and recombines the population to create another set of parent points $\boldsymbol{v}_{i,g}$.

(c) A new point $\boldsymbol{u}_{i,g}$ (offspring point) is created from $\boldsymbol{\theta}_{i,g} = (\theta^1_{i,g}, \dots, \theta^{2p}_{i,g})$ and $\boldsymbol{v}_{i,g} = (v^1_{i,g}, \dots, v^{2p}_{i,g})$ choosing randomly between their coordinates $\theta^j_{i,g}$ and $v^j_{i,g}$ for every $j = 1, \dots, 2p$.

(d) Finally, if $\mathrm{EI}(\boldsymbol{u}_{i,g}) \geq \mathrm{EI}(\boldsymbol{\theta}_{i,g})$ the algorithm replaces $\boldsymbol{\theta}_{i,g}$ with $\boldsymbol{u}_{i,g}$ in the next generation, otherwise $\boldsymbol{\theta}_{i,g}$ is kept.

Steps (b) through (d) are repeated for $N_T$ generations $g$. The algorithm stops when two convergence criteria are fulfilled: the standard deviation $\sigma_{EN}$ of the population's expected improvement (see Fig. A.2(e)) and the average distance $D$ among the population points (see Fig. A.2(f)) are below a certain threshold (we set $\sigma_{EN} = D = 10^{-3}$). When this happens, the point in the population with the maximum expected improvement is selected as $\tilde{\boldsymbol{\theta}}$. We notice that the criterion on the distance guarantees that in a flat landscape like the one of EI($\boldsymbol{\theta}$) the points do not get stuck on a plateau and concentrate closer to a unique candidate maximum. Although the algorithm requires many evaluations of EI($\boldsymbol{u}_{i,g}$) (as shown also in the main text) it is a valid algorithm for finding the maximum within the flat landscape of the acquisition function.

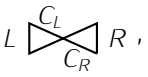
### A.2.2 Other optimizers

*Basin-hopping* – Basin-hopping is a global stochastic optimization algorithm [WD97]. It combines two steps: (i) a local optimization which proposes a candidate solution and (ii) a perturbation of such candidate in order to make it hop to other basins which might contain a global optimal point. The new point is accepted or rejected according to a probability which depends on a "temperature" parameter. The "temperature" parameter decreases with the iteration number so that, at the beginning, new proposals are easily accepted while, at larger iterations, the algorithms becomes more and more selective. The algorithm runs for a fixed number of iterations and the local optimizer used in this context is the gradient based $BFGS$ algorithm.

*Dual annealing* – This global optimization algorithm is the generalized form of the simulated annealing and it is paired with a local optimization which is performed at the end of the annealing to refine the solution [XSFG97]. It is a variation of a hill climbing algorithm in which a solution is randomly perturbed and the new proposed point is accepted with a probability that depends on the difference in energy between the two points. This probability also depends on a "temperature" parameter that, like in the basin-hopping case, decreases with the number of iterations in order to converge to a candidate solution.

# B | Gates of Dimer State MBQC

All the sequences of measurements are applied to a single bow-tie of 6 qubits, and we name its components like this:



(B.1)

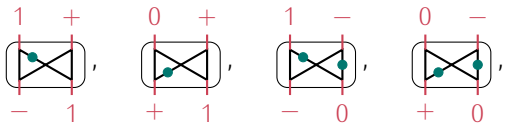
where $L/R$ represent the left/right-most qubit and $C_{L/R}$ represent the central left/right pair of sites.

## B.1 Decoupling Gate

We first need to show how to transport the state of the logical qubits without affecting them. The sequence of measurements on a bow-tie that allows for transport (no operation on the logical qubits) is:

1. Qubits $L$ and $R$ are measured in the $|g\rangle \pm |e\rangle$ basis,
2. Apply phase shift of $\phi = \frac{\pi}{2}$ to the lower qubits of $C_R$ and $C_L$.
3. Perform measurement (III)

This projects the bow-tie on a superposition of states in which only the following four are allowed:

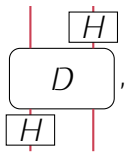


(B.2)

where we are willingly ignoring phases that arise depending on the state measured. Since computation is happening from top to bottom we can see that the following transformation is being applied to the qubits (simply reading the incoming qubit states and the outgoing ones):

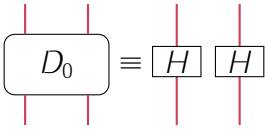
$$
\begin{aligned}
|1+\rangle &\to |-1\rangle \\
|0+\rangle &\to |+1\rangle \\
|1-\rangle &\to |-0\rangle \\
|0-\rangle &\to |+0\rangle \,.
\end{aligned}
$$

To understand what gate is being applied we transform all states into the computational basis by applying $H$:




$$D = \begin{cases} |10\rangle & \to |11\rangle \\ |00\rangle & \to |01\rangle \\ |11\rangle & \to |10\rangle \\ |01\rangle & \to |00\rangle . \end{cases} \tag{B.3}$$

from which it is easy to infer that there is only one $X$ gate being applied to the right qubit. Since we are not interested in residue Pauli gates (that can be later corrected) the action of gate $D_0$ is simply:

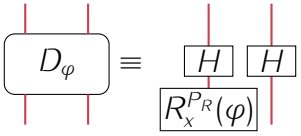



(B.4)

which, a part from a known $H$ operation, does not act on the qubits and can be eliminated with a subsequent application of $H$.

## B.2 Single qubit rotation

In order to apply a single qubit rotation using the scheme proposed by [Cré22], it suffices to add a phase $\psi = \phi$ phase to the left-most $L$ qubit before applying the same sequence of the decoupling gate. This generates the gate




(B.5)

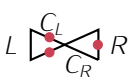
which corresponds to the application of $R_x(\phi)$ to the $L$ qubit according to the measurement of $P_R = 0, 1$. In case $P_R = 0$ the gate is identical to the decoupling gate and the logical information of the qubits flows unchanged. Thus, the chance of applying this gate is 50%.

The non deterministic aspect of this gate can be dealt with by repeating the same measurement scheme until the rotation is applied. A simple argument in [Cré22] shows that this adds a polynomial overhead on the number of measurements. That is due to the fact that the chance of the gate to fail for $k$ consecutive times scales as $\propto \frac{1}{2^k}$.

## B.3 Entangling gate

To apply entanglement one possible sequence is:

1. Dephase by an angle $\pi$ (denoted by a red dot) the following three qubits:




(B.6)

which means adding the phase only to the $|1\rangle$ state, that means we now have the basis elements $\{\triangleright, \triangleright, -\triangleright, -\triangleright\}$ on the left and $\{\triangleleft, -\triangleleft, \triangleleft, \triangleleft\}$

2. Now apply measurement scheme (II) to obtain:

$$\begin{aligned}&\left(\triangleright - p^L_\uparrow p^L_\downarrow \triangleright + x^L p^L_\uparrow \triangleright + x^L p^L_\downarrow \triangleright\right)\\ &\quad\times\left(\triangleleft + p^R_\uparrow p^R_\downarrow \triangleleft - x^R p^R_\uparrow \triangleleft - x^R p^R_\downarrow \triangleleft\right),\end{aligned} \tag{B.7}$$

which is just the normal action of measurement (II) but adding the pulses signs.

3. Each of the physical tensors "survives" this step and we found a state:

$$-x^R p^R_\downarrow \quad \begin{matrix}0 & - \\ - & 1\end{matrix} \quad , \quad -x^R p^R_\uparrow \quad \begin{matrix}0 & + \\ - & 0\end{matrix} \quad , \tag{B.8}$$

$$+x^L p^L_\downarrow \quad \begin{matrix}1 & + \\ - & 1\end{matrix} \quad , \quad +x^L p^L_\uparrow \quad \begin{matrix}0 & + \\ + & 1\end{matrix} \tag{B.9}$$

$$+p^L_\uparrow p^L_\downarrow x^R p^R_\downarrow \quad \begin{matrix}1 & - \\ + & 1\end{matrix} \quad , \quad +p^L_\uparrow p^L_\downarrow x^R p^R_\uparrow \quad \begin{matrix}1 & + \\ + & 0\end{matrix} \quad , \tag{B.10}$$

$$+p^R_\uparrow p^R_\downarrow x^L p^L_\downarrow \quad \begin{matrix}1 & - \\ - & 0\end{matrix} \quad , \quad +p^R_\uparrow p^R_\downarrow x^L p^L_\uparrow \quad \begin{matrix}0 & - \\ + & 0\end{matrix} \tag{B.11}$$

As before, we now collect the ones with the same input and call this operation $Q$:

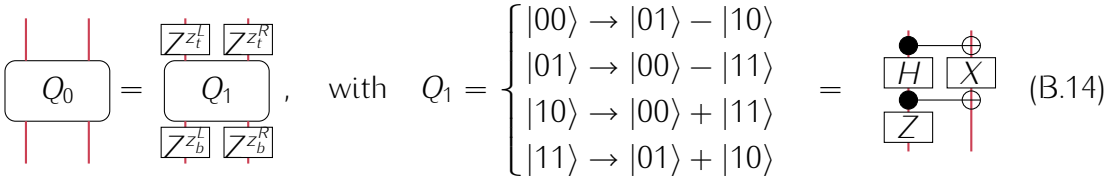
$$Q = \begin{cases} |0+\rangle \to -x^R p^R_\uparrow |-0\rangle + x^L p^L_\uparrow |+1\rangle \\ |0-\rangle \to -x^R p^R_\downarrow |-1\rangle + p^R_\uparrow p^R_\downarrow x^L p^L_\uparrow |+0\rangle \\ |1+\rangle \to p^L_\uparrow p^L_\downarrow x^R p^R_\uparrow |+0\rangle + x^L p^L_\downarrow |-1\rangle \\ |1-\rangle \to p^L_\uparrow p^L_\downarrow x^R p^R_\downarrow |+1\rangle - p^R_\uparrow p^R_\downarrow x^L p^L_\downarrow |-0\rangle \end{cases} \tag{B.12}$$

Visually we absorb the action of $H$ at the beginning and end of gate $Q$ and call it $Q_0$:

$$Q = H\ Q_0\ H, \quad \text{with} \quad Q_0 = \begin{cases} |00\rangle \to -x^R p^R_\uparrow |10\rangle + x^L p^L_\uparrow |01\rangle \\ |01\rangle \to -x^R p^R_\downarrow |11\rangle + p^R_\uparrow p^R_\downarrow x^L p^L_\uparrow |00\rangle \\ |10\rangle \to p^L_\uparrow p^L_\downarrow x^R p^R_\uparrow |00\rangle + x^L p^L_\downarrow |11\rangle \\ |11\rangle \to p^L_\uparrow p^L_\downarrow x^R p^R_\downarrow |01\rangle - p^R_\uparrow p^R_\downarrow x^L p^L_\downarrow |10\rangle \end{cases} \tag{B.13}$$

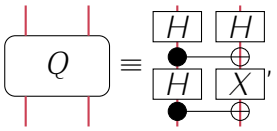
Which can be cheked directly (we just removed the action of $H$ on the left qubit before applying the gate (to the left of the arrow) and to the right-one after applying the gate (to the right of the arrow). Erasing all the phases introducted by Pauli

errors we are left with $Q_1$:




(B.14)

By shifting the lower $H$ of $Q$ to the top and leaving behind the long string of Pauli errors we are left with:




(B.15)