Alma Mater Studiorum · University of Bologna

School of Science
Department of Physics and Astronomy
Master Degree in Physics

Topology-Preserving Embedding of Maximum Independent Set Instances for Rydberg-Atom Quantum Optimizers

Supervisor:
Prof. Elisa Ercolessi

Co-supervisor:
Dr. Federico Dell'Anna

Submitted by:
Riccardo Vidali

Academic Year 2024/2025

Abstract

This work investigates the resolution of a well-known NP-complete problem, the maximum independent set (MIS) problem, in the setting of two-dimensional unit-disk graphs (UDGs). MIS is a paradigmatic hard problem with applications ranging from network design and resource allocation to coding theory, and serves as a standard benchmark for both classical and quantum optimisation methods. In our approach, we first reformulate MIS as a constrained quadratic unconstrained binary optimization (QUBO) problem, obtaining a three-dimensional interaction graph.

We then introduce a geometric embedding algorithm that maps this three-dimensional instance to a two-dimensional UDG with only nearest-neighbour interactions, while being topologically equivalent to the original interaction graph (i.e. preserving its connectivity), thus making it compatible with neutral-atom quantum devices based on Rydberg arrays. We analyse the performance of this embedding procedure in terms of its success probability and of the scaling of the final graph size with the original system size and connectivity.

Finally, we tackle the MIS problem on the embedded graphs using a variational hybrid quantum-classical algorithm of the quantum approximate optimization algorithm's (QAOA) family, and compare its performance to a classical brute-force solver. This allows us to characterise the regimes in which such quantum-inspired approaches can effectively handle MIS instances arising from unit-disk graphs under realistic hardware constraints.

Sommario

Questo lavoro studia la risoluzione di un famoso problema NP-completo, il problema dell'insieme indipendente massimale (MIS), nel contesto dei grafi a disco unitario bidimensionali. Il MIS è un problema paradigmatico di elevata complessità, con applicazioni che spaziano dalla progettazione di reti e dall'allocazione di risorse alla teoria dei codici, e rappresenta un banco di prova standard sia per metodi di ottimizzazione classici sia quantistici. Nel nostro approccio, riformuliamo innanzitutto il MIS come un problema QUBO (quadratic unconstrained binary optimization) vincolato, ottenendo così un grafo di interazione tridimensionale.

Introduciamo quindi un algoritmo di integrazione geometrica che mappa questa istanza tridimensionale in un UDG bidimensionale con sole interazioni tra primi vicini, mantenendo al contempo l'equivalenza topologica con il grafo di interazione originale (cioè preservandone la connettività), rendendola così compatibile con dispositivi quantistici a atomi neutri basati su registri di stati di Rydberg. Analizziamo le prestazioni di questo algoritmo in termini di probabilità di successo e di scalabilità della dimensione finale del grafo rispetto alla dimensione e alla connettività del sistema originale.

Infine, affrontiamo il problema MIS sui grafi finali utilizzando un algoritmo ibrido variazionale quantistico-classico della famiglia QAOA (quantum approximate optimization algorithm), e ne confrontiamo le prestazioni con quelle di un risolutore classico di forza bruta. Ciò ci permette di caratterizzare i regimi in cui tali approcci di ispirazione quantistica possono trattare efficacemente istanze di MIS derivanti da grafi a disco unitario, sotto vincoli hardware realistici.

Contents

Introduction

Mathematical optimization (or mathematical programming) concerns the problem of selecting the best element, according to some given criterion, from a set of admissible alternatives. From a modern perspective, one usually distinguishes between *continuous optimization*, where the variables range over real domains, and *discrete optimization*, where the search space is combinatorial. Optimization problems appear in a variety of different disciplines, from engineering and computer science to operations research and economics, and the search for efficient solution methods has been a central theme in applied mathematics for centuries.

Historically, the calculus-based foundations of optimization were laid by Fermat [1] and Lagrange [2], who derived conditions for extrema of real-valued functions, while Newton [3] and Gauss [4] developed iterative procedures to approach optimal points numerically. However, the modern version of optimization problems came only with the emergence of *linear programming*, in the 1940s, developed by Kantorovich [5] and Dantzig [6]. Together, these developments shaped modern optimization theory and established linear and discrete optimization as core tools in both mathematics and the applied sciences.

Among the many classes of optimization problems, those whose difficulty grows combinatorially with the system size have acquired particular prominence since the 1970s, with the development of computational complexity theory. In this framework, a central role is played by the class of **NP-complete** problems: decision problems for which a proposed solution can be verified efficiently, but for which no efficient algorithm is known to find a solution in the worst case. Seminal works by Cook [7], Karp [8] and others [9] showed that a wide variety of seemingly unrelated combinatorial tasks can be reduced to one another and are therefore equivalent in difficulty.

These NP-complete problems has grown a progressively increasing interest over the years, since many real-world tasks, such as scheduling, routing, resource allocation, and network design, can be formulated as NP-complete optimization problems, so progress on these cases can translate directly into practical gains in efficiency and cost. At the same time, they are difficult to solve because the naive search space typically grows exponentially with the problem size, and no polynomial-time algorithms are known despite decades of effort. As a result, even moderately sized instances can quickly exceed the capabilities of brute-force or straightforward exact methods, prompting the development of sophisticated heuristics, approximation algorithms, and alternative computational paradigms.

In light of this intrinsic hardness, a major line of research in the last decades has been to explore **quantum computation** as a new paradigm for tackling this type of problems. The hope is that genuinely quantum resources, e.g. superposition and entanglement, might enable algorithms that outperform the best classical methods, at least for certain structured instances or in approximate settings. Although there is strong evidence that generic NP-complete problems cannot be solved efficiently even on a quantum computer, the search for quantum speed-ups has led to a rich family of heuristic and variational algorithms that are now routinely tested on combinatorial optimisation tasks.

A prominent example of this line of research is the **quantum approximate optimization algorithm** (**QAOA**), introduced by Farhi, Goldstone and Gutmann in 2014 [10] as a variational approach to combinatorial optimisation on near-term devices. In its original formulation, QAOA alternates between the application of a "problem" Hamiltonian encoding the *cost function* and a "mixing" Hamiltonian, with a small number of variational layers whose parameters are optimized classically to minimise the expected cost. Since its proposal, QAOA has become a standard benchmark for **noisy intermediate-scale quantum** (**NISQ**) [11] hardware and a central testbed for studying quantum advantage in approximate optimisation, giving rise to numerous extensions and a large body of analytical and numerical work on its performance for specific problem classes.

A particularly relevant example of an NP-complete optimization problem is finding the **maximum independent set** (MIS) of a **unit-disk graph** (**UDG**). This problem can be naturally encoded on **Rydberg-atom** quantum computers, thanks to the structure of atomic interactions in the *blockade regime* (Sec. 2.1).

In the specific context of this work, the MIS problem is first mapped to a **quadratic unconstrained binary optimization** (**QUBO**) problem, to which we then add additional constraints targeting particular graph structures. These constraints, implemented as auxiliary nodes, render the corresponding interaction graph three-dimensional. However, most current Rydberg-atom quantum devices can only realize two-dimensional arrays of atoms. To overcome this mismatch, we develop a geometric embedding algorithm whose purpose is to map these three-dimensional instances to topologically equivalent two-dimensional instances with strictly nearest-neighbor connectivity.

This work is organized as follows. In chapter 1 we introduce the basic notions of mathematical optimization and the formalism of QUBO problems. In chapter 2 we review the essential physics of Rydberg atoms and neutral-atom platforms for quantum computing. In chapter 3 we formulate the MIS problem in the specific context considered in this thesis and discuss how it gives rise to a three-dimensional QUBO interaction graph; we also propose a classical embedding algorithm to solve the geometric embedding problem that naturally appears when recasting the original problem in QUBO form. In chapter 4 we then introduce the QIRO algorithm [12] as the variational quantum approach used to solve the embedded instances, and we analyse the performance of both QIRO (compared to a classical brute-force solver) and the embedding algorithm, in terms of success probability and of the scaling of the final graph size with the original system size and connectivity. Finally, appendix A summarizes the density-operator formalism in quantum mechanics, appendix B reviews the basic notions of complexity theory, and appendix C collects the full set of embedding-simulation results discussed (but not fully reported) in chapter 4.

1 Optimization Problems

1.1 Introduction

In the 1940s, Kantorovich [5] and Dantzig [6] gave the first explicit *modern* definition of an *optimization problem*: maximize or minimize a linear function subject to linear constraints. This formulation, now known as **linear programming**, provided a general mathematical framework that still underlies much of optimization theory today.

Among the multitude of optimization problems, one has achieved particular notoriety: the **Traveling Salesman Problem** (TSP). As Cook emphasizes in his account [13], the TSP quickly became the "popular face" of optimization: easy to state, notoriously hard to solve, and deeply relevant to real world applications such as routing, logistics, and network design. Decades of progress in mathematics, computer science, and even physics have been driven by attempts to tackle the TSP, making it both a symbol of computational intractability and a catalyst for methodological innovation.

An important property of optimization problems, emphasized by Papadimitriou [14], is that any such problem can be reformulated as an essentially equivalent decision problem (Def. B.5) by introducing a threshold value for the objective function and asking whether there exists a feasible solution whose value meets or exceeds this threshold.

Optimization problems arise in both *discrete* and *continuous* forms, with applications ranging from graph theory to physics and engineering. Their computational difficulty varies widely: some are solvable efficiently, others are intractable without approximation. This diversity of structure and complexity makes optimization a natural bridge between abstract computational theory and real-world applications.

1.2 Classes of Optimization Problems

To study these problems systematically, it is useful to introduce a more formal framework, and thus we start with the general definition of an **optimization problem** [15].

Def. 1.1. An optimization problem takes the form

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \le b_i, \quad i = 1, \ldots, m \end{array} \tag{1.1}$$

The vector $x \in \mathbb{R}^n$ is the **optimization variable**, the function $f_0 : \mathbb{R}^n \to \mathbb{R}$ is the **objective function** or **cost function**, the inequalities $f_i(x) \le 0$ are called **inequality constraints**, the functions $f_i : \mathbb{R}^n \to \mathbb{R}$ are the **inequality constraint functions**, and the constants $b_1, \ldots, b_m$ are the limits, or **bounds**, for the constraints.

Equivalently, each constraint can be written as $f_i(x) - b_i \le 0$, so the canonical form uses $f_i(x) \le 0$, which is more convenient. We will adopt this convention in what follows.

While inequality constraints already cover many situations, in practice optimization problems often involve requirements that must hold exactly, such as balance equations or conservation laws. To account for these, it is convenient to extend the framework to include equality constraints, leading to the following standard form [15].

Def. 1.2. An optimization problem has the form

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \le 0, \quad i = 1, \dots, m \\ & h_j(x) = 0, \quad j = 1, \dots, p \end{array} \tag{1.2}$$

The equations $h_j(x) = 0$ are the **equality constraints** and the functions $h_j : \mathbb{R}^n \to \mathbb{R}$ are the **equality constraint functions**. If there are no constraints (i.e., $m = p = 0$) we say the problem (1.2) is **unconstrained**.

To fully describe an optimization problem, we must specify what it means to *solve* it. The **domain** is the set of points where all objective and constraint functions are defined. A point is **feasible** if it satisfies all constraints, and the set of such points is the *feasible set* (or *constraint set*). If this set is empty, the problem is *infeasible*; otherwise, it is *feasible*. Feasible points that *minimize the objective function* are **optimal solutions**, and the corresponding objective value is the **optimal value**:

$$f_0(x^\star) = p^\star = \inf \left\{ f_0(x) \mid f_i(x) \le 0,\ i = 1, \dots, m,\ h_i(x) = 0,\ i = 1, \dots, p \right\}. \tag{1.3}$$

We will then call a **solution method** for an optimization problem any algorithm that, given a specific *instance* of the problem, computes a *solution* to within a prescribed accuracy. By convention we focus on *minimization problems*, since any *maximization problem* can be written in this form by minimizing the function $-f_0(x)$ subject to the same constraints. Thus, all notions above extend directly to maximization, where the optimal value is defined as a *supremum* rather than an *infimum*.

We generally study *families* or *classes* of optimization problems, distinguished by the specific forms of their objective and constraint functions. As a first example, we consider *linear programming*, already introduced above. Formally, it is defined as follows [15]:

Def. 1.3. When objective and constraint functions are all *affine*, the problem is called a **linear program** (LP), which has the form

$$\begin{array}{ll} \text{minimize} & c^T x + d \\ \text{subject to} & Gx \le h \\ & Ax = b \end{array} \tag{1.4}$$

where $G \in \mathbb{R}^{m \times n}$ and $A \in \mathbb{R}^{p \times n}$. If the objective or some constraints are not affine, the problem is called a **nonlinear program**. We recall that a function $f : \mathbb{R}^n \to \mathbb{R}$ is *affine* if it can be written as $f(x) = a^T x + b$ for some $a \in \mathbb{R}^n$, $b \in \mathbb{R}$. If $b = 0$, the function is (strictly) *linear*.

We also notice that, it is customary to omit the constant term d in the objective function (1.4), as it does not affect the feasible set. Furthermore, the inequality $Gx \le h$ is understood componentwise, meaning that $(Gx)_i \le h_i$ for all i.

Linear programs are in fact a special case of a broader and particularly important class, namely **convex optimization problems**, which we introduce next [15].

Def. 1.4. A convex optimization problem is one in which the objective and constraint functions are *convex*[1], which means they satisfy the inequality

$$f_i(\alpha x + \beta y) \leq \alpha f_i(x) + \beta f_i(y) \tag{1.5}$$

for all $x, y \in \mathbb{R}^n$ and all α, $\beta \in \mathbb{R}$ with $\alpha + \beta = 1$, $\alpha \geq 0$, $\beta \geq 0$. In particular, a convex optimization problem, has the form

$$\begin{array}{lll} \text{minimize} & f_0(x) & \\ \text{subject to} & f_i(x) \leq 0, & i = 1, \dots, m \\ & h_j(x) = a_j^T x = 0, & j = 1, \dots, p \end{array} \tag{1.6}$$

Comparing (1.6) with the general standard form problem (1.1), we immediately notice that the convex problem has three additional requirements:

- The objective function must be convex.
- The inequality constraint functions must be convex.
- The equality constraint functions $h_j(x)$ must be affine.

Extending the concept, a function is called **quasiconvex** if all its sublevel sets $\{x \mid f(x) \leq \alpha\}$ are convex. A function is instead **concave** if it satisfies the reversed inequality

$$f_i(\alpha x + \beta y) \geq \alpha f_i(x) + \beta f_i(y) \tag{1.7}$$

and it is **quasiconcave** if all its superlevel sets $\{x \mid f(x) \geq \alpha\}$ are convex. Accordingly, optimization problems are termed convex, quasiconvex, concave, or quasiconcave depending on the nature of their objective and constraint functions.

Beyond convex optimization, an important class is given by **integer programs** (IP) [16], where some or all of the variables are constrained to take integer values, leading to fundamentally different, and generally much harder[2], problems.

Def. 1.5. An integer program has the form

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & Ax \leq 0 \end{array} \tag{1.8}$$

where $A \in \mathbb{Z}^{m \times n}$ and $c, x \in \mathbb{Z}^n$. If integrality constraints are imposed only on a subset of the variables, the problem is called a **mixed integer program** (MIP).

[1] Convexity generalizes linearity: the equality is relaxed to an inequality, required only for convex combinations. Moreover, both the *domain* and the *feasible set* of a convex problem are convex.

[2] The feasible set of an integer program is *nonconvex*, so the geometric methods of LP no longer apply. Moreover, many NP-hard problems can be formulated as IPs, making efficient algorithms unlikely.

While IP already captures a broad class of problems where the feasible region is restricted to discrete sets, the scope of **combinatorial optimization** is even more general. In this setting, one instead considers *arbitrary finite sets of feasible solutions* defined by combinatorial structures (for example graphs). To formalize this idea, we introduce the general notion of a **NP-optimization problem** [16]:

Def. 1.6. An NP-optimization problem has the form

$$\begin{aligned} \text{minimize} \quad & c(x, y) \\ \text{subject to} \quad & x \in X \\ & y \in S_x \end{aligned} \tag{1.9}$$

Here, X is the *set of problem instances*, each encodable as a binary string and decidable in polynomial time. S_x is the *set of feasible solutions* for instance $x \in X$, with membership decidable in polynomial time. $c : (x, y) \to \mathbb{Q}$ is the cost function computable in polynomial time[3].

As already noted, complexity classes are defined in terms of *decision problems* (Def. B.5). For this reason, every optimization problem has an associated decision version, and its computational complexity is usually analyzed through that version.

In the context of optimization problems, the notion of *NP-hardness* again indicates that no polynomial-time algorithm is expected to exist, unless $P = NP$. To refine this picture, three further classifications [16] are useful.

Def. 1.7. A problem that polynomially reduces to some problem in NP is called **NP-easy**[4]. If a problem is both NP-hard and NP-easy, it is called **NP-equivalent**. A problem is **strongly NP-hard** if it remains NP-hard even when all numerical parameters in the input are bounded by a polynomial in the input size.

Moreover, the following result can be established [16], showing that every NP-optimization problem reduces in polynomial time to its associated NP decision problem.

Th. 1.1. Every NP-optimization problem is NP-easy.

Examples help to illustrate these classifications. Problems such as the *optimization version* of the TSP are NP-easy, since they can be reduced to their decision counterpart in NP. Many natural optimization problems, including TSP optimization and the MIS problem, are in fact both NP-hard and NP-easy, and are therefore NP-equivalent. On the other hand, problems like *3-Partition* [9, 17] are strongly NP-hard, as they remain intractable even when numerical parameters are bounded by a polynomial in the input size. Within this landscape, the *Weighted MIS problem* which will be treated in this work is a canonical example of an *NP-equivalent combinatorial optimization problem.*

[3] While many problems (e.g. integer programs) have integer costs, the standard definition uses $\mathbb{Q}$ for generality, covering also objectives involving ratios or normalizations.

[4] Note that NP-easy does not imply efficient (polynomial-time) solvability; it only means the problem is no harder than some problem in NP.

1.3 Slack Variables

In what follows, **slack variables** will play a central role. They serve as auxiliary variables that transform inequalities into equalities, thereby simplifying the analysis and facilitating the application of standard optimization methods. We now proceed to define them [15].

Def. 1.8. The inequality constraints $f_i(x) \leq 0$ can be reformulated as equalities by introducing additional (slack) variables $s_i \geq 0$ such that $f_i(x)+s_i = 0$. Since each f_i is a scalar-valued function, a single scalar slack variable is sufficient for each constraint. Using this transformation we obtain the problem

$$\begin{array}{lll} \text{minimize} & f_0(x) & \\ \text{subject to} & s_i \geq 0, & i = 1, \dots, m \\ & f_i(x) + s_i = 0, & i = 1, \dots, m \\ & h_j(x) = 0, & j = 1, \dots, p \end{array} \tag{1.10}$$

where the variables are $x \in \mathbb{R}^n$ and $s \in \mathbb{R}^m$.

The reformulated problem (1.10) involves $n+m$ variables, m nonnegativity constraints on the slack variables, and $m+p$ equality constraints. In other words, each original inequality is replaced by an equality constraint together with a simple nonnegativity condition.

Moreover, the problem (1.10) is equivalent to the original problem (1.1). Indeed, if (x, s) is feasible for (1.10), then $s_i = -f_i(x) \geq 0$ implies that x is feasible for the original problem. Conversely, any feasible x for (1.1) yields a feasible pair (x, s) with $s_i = -f_i(x)$. The same correspondence clearly holds also for optimal solutions. Importantly, the introduction of slack variables does not alter the structural properties of the problem (for instance, convexity is preserved when applied to linear inequalities).

At an optimal solution, slack variables also indicate which constraints are *active*[5]: a zero slack means the inequality is tight (i.e., the inequality holds with equality at the optimal point), while a positive slack shows that the constraint is not binding [18].

Just as (1.2) provides a *standard form* for the general optimization problem (1.1), LPs 1.3 also admit a standard formulation, commonly referred to as the **equational form** [19].

Def. 1.9. A LP in equational form is given by

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & Ax = b \\ & x \geq 0 \end{array} \tag{1.11}$$

where $x, c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ and $A \in \mathbb{R}^{m \times n}$. Here, the zero symbol denotes the vector $0 \in \mathbb{R}^n$, so that the inequality is understood componentwise, i.e., $x_i \geq 0$ for all i.

It can be shown that, by introducing slack variables, any linear program can be reduced to the equational form (1.11). This representation is usually preferred because it expresses all constraints in a uniform way (equalities with nonnegative variables) and serves as the natural input format for algorithms such as the *simplex method* [18, 19].

[5] A constraint is active at a solution if it holds with equality, and inactive if it is satisfied strictly.

1.4 QUBO Problems

Studies on the **quadratic unconstrained binary optimization problem** (QUBO) can be traced back to the 1960s, when there was much work on the class of *pseudo-Boolean optimization problems* [20, 21]; while the modern QUBO terminology was popularized in the 2000s by Glover and Kochenberger [22, 23] as a unifying framework for a wide class of combinatorial optimization problems. We now give its formal definition [24].

Def. 1.10. QUBO is defined as the problem

$$\begin{aligned} &\text{minimize} && x^T Q x + c^T x \\ &\text{subject to} && x \in \{0,1\}^n \end{aligned} \tag{1.12}$$

where $Q \in \mathbb{R}^{n\times n}$ is a symmetric matrix of quadratic coefficients and $c \in \mathbb{R}^n$ is a vector of linear coefficients.

More precisely, the QUBO problem (1.12) is often formulated as a maximization task. However, as already said, since maximization and minimization are trivially equivalent up to a global sign change of the objective function, we will stick with the minimization form. Furthermore, the matrix Q can be assumed symmetric[6] without loss of generality, since only its symmetric part contributes to the quadratic form [25]:

$$x^T Q x \longrightarrow x^T \frac{Q + Q^T}{2} x \tag{1.13}$$

Componentwise this corresponds to the substitution

$$Q_{ij} \longrightarrow \frac{Q_{ij} + Q_{ji}}{2} \tag{1.14}$$

Many classical *graph-theoretic optimization problems*, such as the *maximum weight stable set* problem, the *maximum weight clique* problem, and the *maximum cut* problem, can be reformulated as QUBO instances. This correspondence reveals a hidden QUBO structure underlying a broad class of combinatorial problems, linking the history of QUBO to the development of these graph-based formulations [24].

Although the natural QUBO form is unconstrained apart from the binary restriction on variables, many practical optimization problems involve additional constraints. These can often be incorporated directly into the *quadratic objective* through suitable *penalty terms*, chosen to vanish for feasible solutions and to take positive values otherwise. In this way, *constrained problems* can be reformulated as *unconstrained QUBO instances* [25].

Moreover, QUBO models can also be expressed as *continuous* optimization problems through suitable reformulations, such as quadratic or semidefinite programs [26, 27].

More precisely, the QUBO formulation first appeared in statistical mechanics under an alternative representation [28] known as the **Ising model** [29]. In this model, the variables take values in $\{-1, 1\}$, and it was originally proposed in the 1920s by Wilhelm Lenz [30] and Ernst Ising [31] to describe the behavior of magnetic materials.

[6] Sometimes, Q is taken in *upper-triangular form* (i.e. all entries below the main diagonal are zero, namely $Q_{ij} = 0\ \forall\, i > j$) to avoid duplicate terms, though the symmetric convention is more common.

The model is mathematically described by the following **Ising Hamiltonian** [24]:

$$H(s) = \sum_{i,j} J_{ij} s_i s_j + \sum_j h_j s_j = s^T J s + h^T s \tag{1.15}$$

where $s_i \in \{-1, 1\}$ represents the *spin variable* of site i, $J \in \mathbb{R}^{n\times n}$ encodes *pairwise interaction strengths*, and $h \in \mathbb{R}^n$ represents *local external fields* acting on each spin. The goal is to find the spin configuration that *minimizes the total energy*, corresponding to the **ground state** of the system. This leads to the following optimization problem [24]:

Def. 1.11. The **Ising QUBO** problem is defined as

$$\begin{aligned} \text{minimize} \quad & s^T J s + h^T s \\ \text{subject to} \quad & s \in \{-1, 1\}^n \end{aligned} \tag{1.16}$$

A key property of QUBO is its equivalence to the Ising formulation, achieved through the affine transformation $s_i = 1 - 2x_i$, which maps binary variables $x_i \in \{0, 1\}$ to spin variables $s_i \in \{-1, 1\}$. Under this this change, the parameters relate as

$$\begin{aligned} s_i &= 1 - 2x_i \\ J &= \frac{Q}{4} \\ h_i &= -\frac{1}{4}\sum_j (Q_{ij} + Q_{ji}) - \frac{1}{2} c_i \end{aligned} \tag{1.17}$$

up to an additive constant that does not affect the minimizer. Minimizing the QUBO objective (1.12) is thus mathematically equivalent to finding the ground state of the Ising Hamiltonian (1.15). Indeed, the Ising model is a special case of the Ising QUBO. This model has seen numerous extensions and generalizations, as discussed in [32–35].

The key advantage of this equivalence is that many NP-complete and NP-hard problems can be reformulated in terms of an Ising (and thus also a QUBO) model, with at most a cubic number of spins relative to the problem size [36].

In general, QUBO is *strongly NP-hard* (Def. 1.7), making the existence of a polynomial-time algorithm unlikely. However, there are several special cases where the QUBO problem can be solved in *polynomial*, *pseudopolynomial*, or *subexponential*[7] time [24].

Beyond these special cases, practical solution approaches often rely on *approximation* methods, *heuristics*, and *metaheuristics* to find near-optimal solutions within a reasonable computational time. A theoretical overview of these algorithms, including an analysis of their performance and efficiency across different problem structures, can be found in [24].

The QUBO formulation thus offers a versatile framework for expressing combinatorial problems, like the weighted MIS problem. Its equivalence to the Ising model also makes it compatible with quantum optimization approaches. In this work, a variant of the **Quantum Approximate Optimization Algorithm** (QAOA) will be introduced to address the QUBO formulation related to our problem.

[7] Pseudopolynomial means "polynomial in the numeric value of the input", whereas subexponential means "faster than polynomial but slower than exponential", typically $O(2^{n^\delta})$ for $0 < \delta < 1$.

2 Rydberg Atoms Platforms

2.1 Rydberg Atoms Basics

Among the various physical platforms available for the realization of *quantum computers*, we focus on those based on **Rydberg atoms**. This choice is motivated by the nature of the *weighted MIS* problem we aim to address, which naturally maps onto the interaction patterns of Rydberg systems. In what follows, we introduce the main physical properties of Rydberg atoms that make them suitable for quantum information processing.

The origin of Rydberg physics can be traced back to the study of the hydrogen spectrum in the late nineteenth century. In 1885, Johann Balmer [37,38] discovered a simple empirical relation describing the wavelengths of the visible lines of atomic hydrogen, known today as the **Balmer series**. In particular, the wavelengths of the Balmer series of transitions from the $n = 2$ states to higher lying levels are given by [39]

$$\lambda = \frac{bn^2}{n^2 - 4} \tag{2.1}$$

where $b = 3.645 \times 10^{-7}\,m$ is an empirical constant determined from the visible spectrum. A few years later, in 1888, Johannes Rydberg [40] generalized Balmer's formula to describe the entire spectral series of hydrogen-like atoms in terms of the inverse wavelength

$$\frac{1}{\lambda} = R_H \left(\frac{1}{n_1^2} - \frac{1}{n_2^2} \right) \qquad n_2 > n_1 \tag{2.2}$$

where $R_H = 1.09678 \times 10^7\,m^{-1}$ is the **Rydberg constant**[1]. Rydberg further introduced the idea that the observed regularities in other elements could be captured by replacing the integer quantum numbers with fractional effective values, an empirical correction now known as the **quantum defect** δ. The generalized form of Eq. (2.2) thus reads

$$\frac{1}{\lambda} = R_H \left[\frac{1}{(n_1 - \delta_1)^2} - \frac{1}{(n_2 - \delta_2)^2} \right] \tag{2.3}$$

which reduces to Eq. (2.2) when $\delta_{1,2} = 0$, as is the case for hydrogen. This empirical generalization laid the foundation for the modern concept of Rydberg atoms: atoms in which one or more electrons are excited to very high principal quantum numbers n.

The main properties of Rydberg atoms [39, 41, 42] stem from how their physical characteristics scale with the principal quantum number n. As n increases, the mean **orbital radius** grows as $\langle r \rangle \sim n^2 a_0$, with $a_0 \approx 5.29 \times 10^{-11}\,m$ the *Bohr radius*, so that highly excited atoms can reach micrometer sizes (thousands of times larger than ground-state atoms). The **binding energy**, scaling as $1/n^2$, becomes correspondingly smaller, making Rydberg states easily accessible with narrowband lasers.

[1] This is the value for hydrogen's atom only.

Since, the **radiative lifetime** scales approximately as n^3, higher Rydberg levels exhibit longer *coherence times*, which allows coherent manipulations over microsecond timescales and results in more stable and controllable qubits. In addition, their **polarizability** grows roughly as n^7, making these atoms extremely sensitive to external electric fields and easily tunable through the *Stark effect* [43].

Even more striking is how rapidly interatomic interactions grow with excitation level. Two Rydberg atoms can couple through either **resonant dipole-dipole interactions**, which dominate at short separations and scale as $1/r^3$, or through **van der Waals forces**, which prevail at larger distances and scale as $1/r^6$. The corresponding interaction coefficients C_3 and C_6 increases roughly respectively as n^4 and n^{11}, so that neighboring atoms spaced several micrometers apart still experience energy shifts of several megahertz.

These strong interatomic interactions enables the so-called **Rydberg blockade**, whereby the excitation of one atom shifts the energy levels of its neighbors by more than the excitation linewidth, preventing multiple excitations within a certain radius

$$R_b \sim \sqrt[6]{\frac{C_6}{\hbar\Omega}} \tag{2.4}$$

with Ω the *Rabi frequency* of the excitation laser. This mechanism naturally realizes the constraint structure of many combinatorial optimization problems (like the MIS), where simultaneous excitations of nearby sites are forbidden.

The Rydberg blockade underlies the implementation of *two-qubit entangling gates*. When a *control* and a *target* atom are both coupled to the same Rydberg state, excitation of the control shifts the target's energy, preventing its excitation and inducing a conditional phase. This mechanism realizes a *controlled-Z (CZ) gate* [41], illustrated in Fig. 2.1.

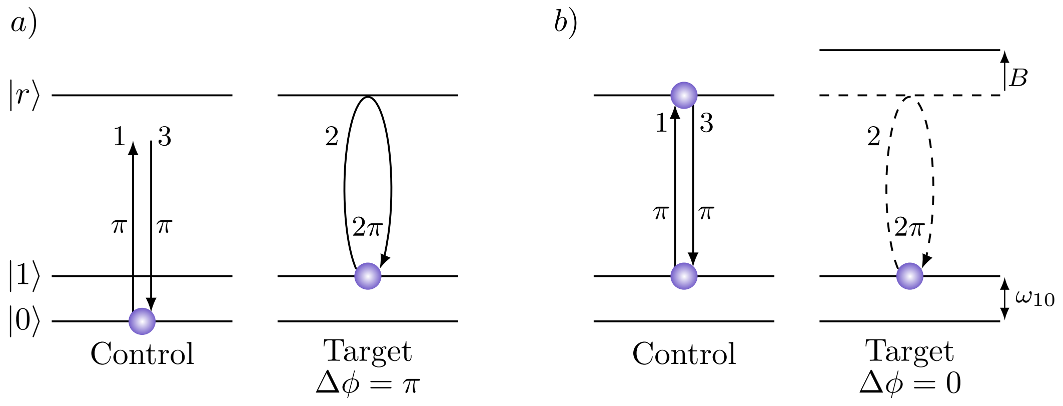


Figure 2.1: *Rydberg blockade CZ gate operating on input states a) $|01\rangle$ and b) $|11\rangle$. The gate sequence consists of three laser pulses: a π-pulse on the control atom, a 2π-pulse on the target, and a final π-pulse on the control. a) When the blockade is inactive, the control atom remains in the ground state and the target acquires a π phase shift. b) Under blockade, the Rydberg level of the target is shifted by B, suppressing its excitation so that no phase shift is accumulated.*

Moreover, the strong tunability of Rydberg interactions, combined with long coherence times and the capability of individual optical control in atom arrays, provides an ideal setting for implementing both **analog** and **digital** quantum computation schemes.

Regarding the digital approach, it is worth spending a few more words on *single-qubit gates* and how they can be implemented using Rydberg atoms. We have already discussed an example of a two-qubit gate (Fig. 2.1), where the meaning of the "π" and "2π" symbols was taken for granted. These refer to **pulses**, whose physical realization and mathematical interpretation we now turn to examine in more detail.

Formally, single-qubit gates are most intuitively understood within the Bloch sphere representation (Sec. A.3), where each gate corresponds to one or more rotations on the sphere. To be more precise, as discussed below, any single-qubit gate can be decomposed into *at most* three rotations about two fixed axes, with the choice of axes not being unique.

As illustrated in Fig. 2.2, the Hadamard gate (A.54), for instance, can be viewed either as two successive rotations, a rotation by $\pi/2$ about the y-axis followed by one by π about the x-axis, or equivalently as a single rotation about the axis (highlighted in light blue in Fig. 2.2b) defined by

$$\hat{n} = \frac{1}{\sqrt{2}}(\hat{x} + \hat{z}) \tag{2.5}$$

From Fig. 2.2, it is evident that a more efficient way to implement the Hadamard gate starting from the $|0\rangle$ state is by performing a single $\pi/2$ rotation about the y-axis.

Conversely, the Pauli gates (A.27) can each be interpreted as a rotation by π about their respective axes, with the Y gate introducing a complex phase and the Z gate applying a phase shift only to the $|1\rangle$ state. In Fig. 2.2, the Z gate is shown as a full 2π loop; this representation is purely illustrative, emphasizing that the $|0\rangle, |1\rangle$ states, being eigenstates of the Z operator and aligned along the z-axis, remain unchanged under its action.

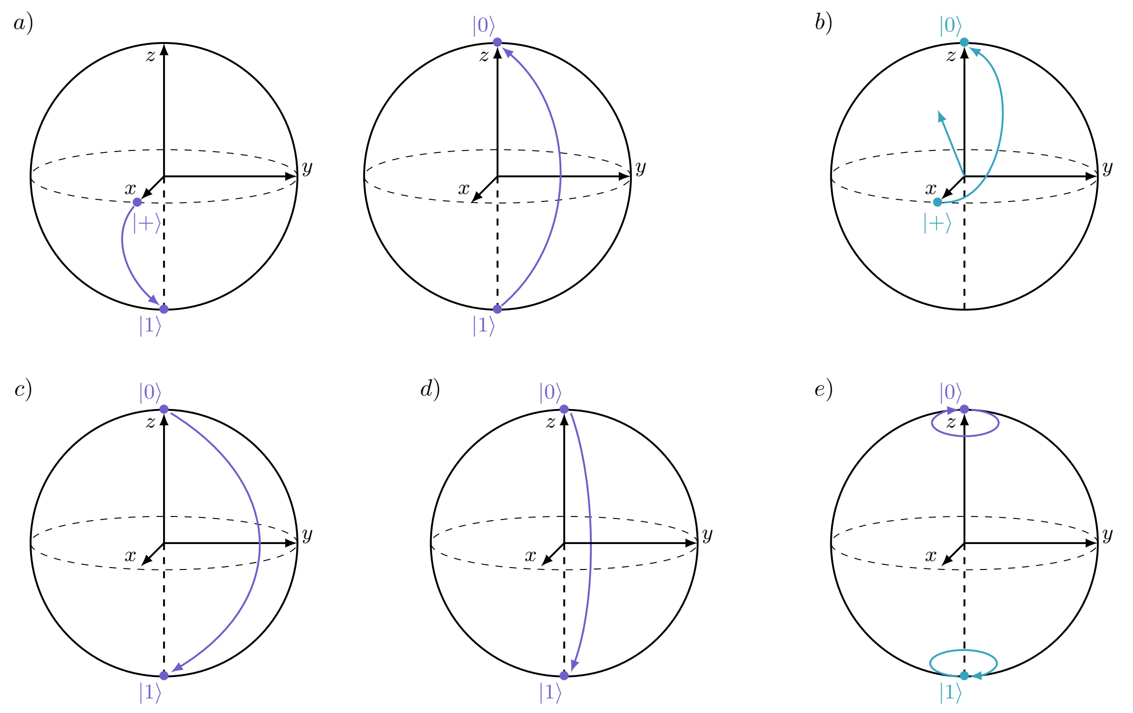


Figure 2.2: *a)–b) Hadamard gate represented via its two equivalent rotation forms. c)–e) Pauli X, Y and Z gates shown as single rotations about their respective axes.*

In general, any single-qubit unitary can be expressed, up to a global phase, as a product of at most three rotations about two distinct axes (*Euler decomposition*) [44], e.g.

$$U = e^{i\alpha} R_z(\beta) R_y(\gamma) R_z(\delta), \tag{2.6}$$

with

$$R_j(\theta) = \exp\left(-i\frac{\theta}{2}\sigma_j\right) \tag{2.7}$$

with $j = x, y, z$. Having understood the mathematical interpretation of the single-qubit gates, we not turn our attention in how these gates, together with two- or multi-qubit gates, are physically performed within the Rydberg atoms' platforms.

In the digital approach one typically encodes the $|0\rangle, |1\rangle$ states in the two hyperfine ground states of the atom, because they have very long lifetimes that prevent radiative coupling to the electromagnetic environment. The quantum gates are then realized by shining fine-tuned laser pulses onto a chosen subset of individual atoms in the register. In particular, specific qubits can be addressed with high accuracy by strongly focusing the lasers [45].

The atom-laser interaction is characterized by the Rabi frequency Ω (proportional to the amplitude of the laser field), the detuning δ (i.e. the difference between the qubit resonance frequency and the actual laser frequency) and their relative phase ϕ. By carefully tuning all these quantities, together with the duration τ of the laser pulse, one can entail a local single qubit rotation about the (x, y, z) axis of angles $(\Omega\tau\cos\phi, \Omega\tau\sin\phi, \delta\tau)$, allowing to realize every possible rotation on the Bloch sphere [46].

Moreover, neutral atom devices present the advantage of natively implementing multi-qubit gates involving more that 2 qubits, which are instrumental for the efficient implementation of several quantum algorithms, including for example *Grover search* [47] or the *variational resolution* of non-linear partial differential equations [48]. For example, it is possible to realize a *three-qubit Toffoli gate* (Fig. A.6), with a sequence of only 7 pulses: 2 Hadamard gates on both sides of a 5-pulse CCZ (Fig. 2.3), whose behaviour is analogous to the CZ gate discussed earlier [45].

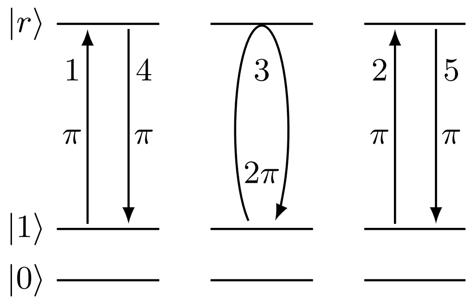


Figure 2.3: *Sequence of five pulses used to realize a CCZ gate.*

We conclude this section by briefly outlining the main future perspective of the digital approach: implement gates with local addressing laser pulses. This requires a considerable hardware improvement, consisting mainly in the massive development of the current capability to address single atoms with lasers.

Moreover, the correction of the main errors, which hinder a perfect implementation of quantum gates, lowering their **fidelity**, has to be taken into account. We recall that the *fidelity* $\mathcal{F}$ of a quantum operation is a number between 0 and 1 that measures the closeness between the state actually created by the operation and the theoretically expected state. For example, a fidelity $\mathcal{F} = 99.5$ % corresponds to a probability of 0.5 % of measuring an undesired outcome after the operation [45,46]. For a comprehensive overview of the latest results on multi-qubit gate fidelities, as well as potential approaches to further improve these fidelities, see [46].

2.2 Rydberg Hamiltonian

In the previous section, we presented various examples of single- and multi-qubit gates implemented with Rydberg atoms within the framework of *digital quantum computing*, where discrete quantum gates are systematically applied to manipulate qubit states. Although the digital approach is, in principle, universal and capable of solving a broader class of problems, it is also considerably more susceptible to errors and therefore requires the use of quantum *error-correction protocols*, as well as larger hardware resources.

In this work, we instead adopt the *analog quantum computing* framework, in which quantum systems evolve continuously under the influence of carefully engineered Hamiltonians, enabling direct simulation of quantum phenomena. This approach is generally less sensitive to errors but also more limited in the class of problems it can efficiently address [49].

To describe the dynamics of such analog systems, we introduce the effective many-body **Rydberg Hamiltonian** describing an ensemble of interacting Rydberg atoms [41,50,51]:

$$\begin{aligned} H(t) =& \frac{1}{2}\hbar\Omega(t)\sum_i \left(e^{i\phi(t)}\left|g_i\right\rangle\!\left\langle r_i\right| + e^{-i\phi(t)}\left|r_i\right\rangle\!\left\langle g_i\right|\right) \\ &- \hbar\Delta(t)\sum_i \left|r_i\right\rangle\!\left\langle r_i\right| + \sum_{i<j}\frac{C_6}{|x_i - x_j|^6}\left|r_i\right\rangle\!\left\langle r_i\right|\left|r_j\right\rangle\!\left\langle r_j\right| \end{aligned} \tag{2.8}$$

For convenience, it is customary to express the Hamiltonian in *dimensionless units* by setting $\hbar = 1$ and choosing a reference phase $\phi(t) = 0$. Defining the *Rydberg-state projector* $n_i = |r_i\rangle\langle r_i|$, the *interatomic distance* $x_{ij} = |x_i - x_j|$, and the *transverse spin operator* $\sigma_i^x = |g_i\rangle\langle r_i| + |r_i\rangle\langle g_i|$, Eq. (2.8) can be written in the more compact form

$$H(t) = \frac{\Omega(t)}{2}\sum_i \sigma_i^x - \Delta(t)\sum_i n_i + C_6\sum_{i<j}\frac{n_i n_j}{x_{ij}^6} \tag{2.9}$$

Here, $\Omega(t)$ is the *Rabi frequency* setting the coherent coupling between $|g\rangle$ and $|r\rangle$, and $\Delta(t)$ is the laser *detuning* from resonance; both are routinely tuned via the laser's intensity and frequency. In Eq. (2.9), the first term drives *Rabi oscillations* [52], the second provides the detuning bias that promotes or suppresses excitation to $|r\rangle$, and the third encodes van der Waals interactions[2] between simultaneously excited atoms. The interplay of drive, detuning, and interaction controls the many-body dynamics of Rydberg ensembles and underpins their use in quantum simulation and optimization.

The Hamiltonian (2.9) can also be expressed in the Ising form via the transformation

$$n_i = \frac{1 + \sigma_i^z}{2} \tag{2.10}$$

which transforms the interaction term $\propto n_i n_j$ into an Ising-type coupling $\propto \sigma_i^z\sigma_j^z$ plus local field terms. This allows the Rydberg Hamiltonian to be rewritten as an effective Ising model [56], directly linking it to classical spin and optimization formulations.

[2] In certain implementations, when two-atom pair states become nearly degenerate, the system enters the *Förster-resonant regime* [53–55], where dipole-dipole interactions (C_3/r_{ij}^3) replace the van der Waals ones (C_6/r_{ij}^6). This regime, tunable via external electric fields, is used for fast entangling gates, while here we operate in the off-resonant blockade regime.

2.3 Platforms for Quantum Computing

Given the nature of our problem, we shall represent qubits in the computational basis $\{|0\rangle, |1\rangle\}$ using the two distinct atomic states [57]:

- We identify $|0\rangle$ with the *ground state* $|g\rangle$ of the atom (usually $5S_{1/2}$)
- We identify $|1\rangle$ with a highly *excited rydberg state* $|r\rangle$ (e.g., $nP_{3/2}$, with $n \gg 1$)[3]

The preparation of Rydberg-based qubits begins with a **magneto-optical trap** (MOT), which combines counter-propagating, red-detuned laser beams with magnetic field gradients to simultaneously cool and confine neutral atoms at microkelvin temperatures. This stage produces a dense and ultracold atomic cloud from which single atoms are subsequently transferred into tightly focused traps for individual control [58–60].

This transfer is achieved using **optical tweezers** [61,62], which use tightly focused laser beams to create localized traps that hold atoms with high precision. These traps allow individual control of each atom, enabling the formation of a **quantum register**, typically a 2D[4] array of Rydberg atoms arranged in arbitrary geometries (Fig. 2.4) [63].

The register's filling is typically verified through **fluorescence imaging** [65–67], where resonant light reveals the presence or absence of atoms in each trap, ensuring full initialization before any *quantum protocol* begins.

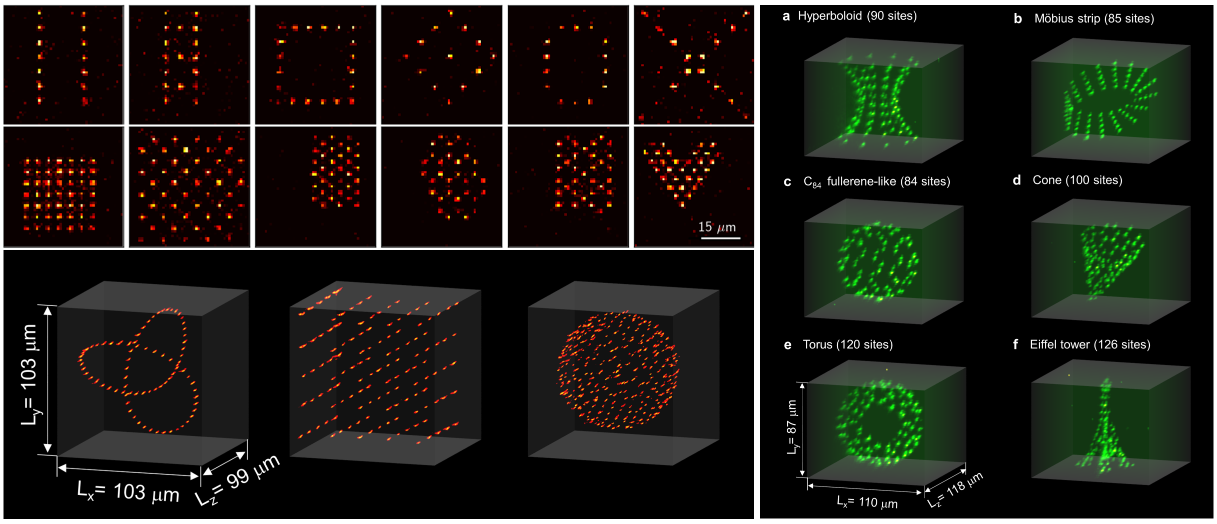


Figure 2.4: *Examples of neutral atom arrays in 2D and 3D* [63,68].

The atoms in the register are typically arranged with *controlled nearest-neighbour spacing*, such that the Rydberg interaction range selectively couples adjacent sites. This geometry enables the study of *many-body dynamics* and *collective excitation phenomena*, while preserving a well-defined interaction topology [56,69,70].

[3] These values refer to Rb^{87} atoms, where the Rydberg states typically lie within the range n between 30 and 100. The $|0\rangle$ state is chosen to be one of the hyperfine ground states, such as $5S_{1/2}, F = 2, m_F = 2$.

[4] Quantum computers based on 3D arrays are also under investigation, though a major challenge remains the individual addressing of atoms for single-qubit gates [64]. In this work, we focus on a 2D quantum register and therefore do not discuss 3D architectures.

Once the atomic register is assembled, *laser fields* provide full control over the internal states and interactions [71]. Different laser configurations can be used to drive *single-photon* [72,73] or *two-photon transitions* [74,75] between the ground and Rydberg states, to coherently couple the $|0\rangle$ and $|1\rangle$ levels, or to induce *collective excitations* through *global pulses* in the *analog* regime [76]. By carefully tuning the *Rabi frequency*, *detuning*, and *phase* of these fields, one can thus manipulate individual qubits, engineer entangling interactions or explore many-body dynamics within the Rydberg array.

Within the analog regime, the laser parameters are often made *time-dependent* to steer the system's evolution continuously. **Adiabatic protocols**, for instance, slowly vary these parameters to keep the system in its instantaneous ground state, enabling robust state preparation and optimization processes. Such time-dependent driving allows the exploration of *quantum phase transitions* and the implementation of *adiabatic quantum algorithms* directly within the Rydberg Hamiltonian [77–81].

At the end of the evolution, a **state-selective fluorescence measurement** distinguishes atoms in the ground and Rydberg states, providing the final *spin configuration* of the system and allowing *direct readout of the computational or many-body outcome* [82,83].

Among the dominant error mechanisms in neutral-atom quantum computing are **leakage**, **crosstalk** and **atom loss**. *Leakage errors* occur when a qubit's population leaves the computational subspace and occupies different energy levels, typically induced by imperfect pulse shaping, off-resonant driving, or interactions with the environment [84,85].

Crosstalk errors arise when control operations on one qubit unintentionally perturb neighboring sites, for example through residual laser illumination, electromagnetic coupling, or shared optical fields [86,87].

Atom loss errors, on the other hand, result from heating, background gas collisions, or anti-trapping effects during Rydberg excitation, leading to missing qubits and reduced register fidelity [70,88,89].

Additional, though less severe, error channels include *decoherence* from finite Rydberg lifetimes, laser *phase noise*, magnetic or electric *field fluctuations*, and *imperfect blockade* interactions, all of which can degrade coherence and gate fidelity.

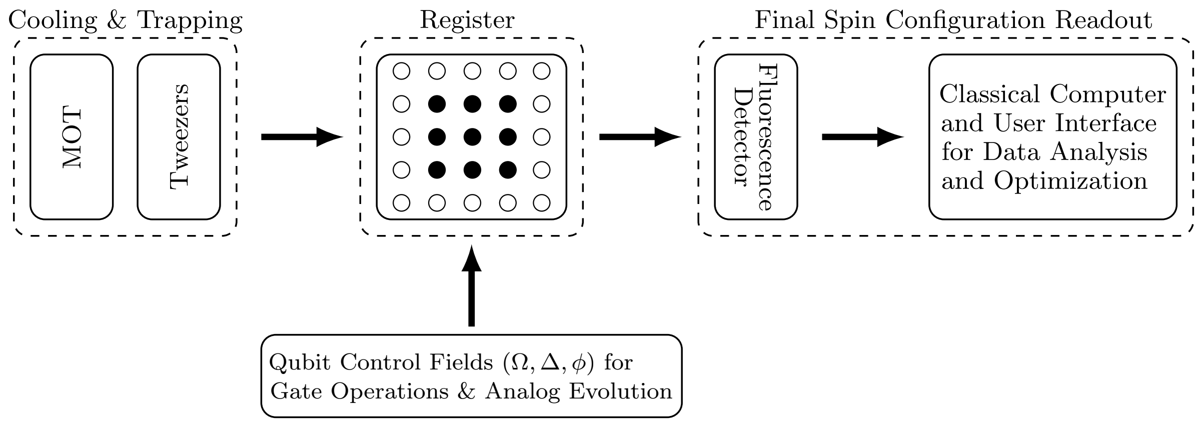

Cooling & Trapping
MOT
Tweezers
Register
Final Spin Configuration Readout
Fluorescence Detector
Classical Computer and User Interface for Data Analysis and Optimization
Qubit Control Fields (Ω, Δ, ϕ) for Gate Operations & Analog Evolution

Figure 2.5: *Schematic overview of a neutral-atom quantum computing platform, showing the main experimental stages: atom cooling and loading, tweezer array formation, qubit control through laser and microwave fields, and fluorescence-based state readout.*

The modular and reconfigurable nature of optical tweezer arrays makes neutral-atom quantum computers inherently *scalable architectures*. By dynamically rearranging traps, merging or splitting arrays, and integrating multiple tweezer planes, these systems can host hundreds of individually addressable qubits while maintaining high connectivity and flexible geometry. For a comprehensive overview of the current state of the art and recent advances in scalability and control, see [90–92].

In the rapidly advancing field of neutral-atom quantum computing, **Pasqal** and **QuEra** stand out as the leading companies in Europe and the United States, respectively. We begin by briefly outlining Pasqal's hardware and technological approachapter

Pasqal's quantum processing units (QPUs) operate in both analog and digital modes, enabling near-term problem-specific simulations and long-term universal quantum computation. The systems run at *room temperature* with a power consumption of about 3 kW and feature up to 140 qubits arranged in configurable 2D arrays within a *modular* architecture (optical core, laser and control units, vacuum and thermal subsystems, and monitoring system). They can be deployed on-premise or accessed via the cloud. Complementing its hardware, Pasqal offers a *Python-based SDK and cloud platform* that allow users to design, simulate, and run circuits or analog pulse sequences on real QPUs.

Pasqal is advancing toward *fault-tolerant quantum computing* (*FTQC*) through hardware progress and collaborations on *quantum-error correction* and *logical-qubit encoding*, aiming for large-scale logical architectures beyond 2027, when the next-generation *Vela* QPU (with more than 200 qubits, improved control and stability) is expected [45, 93].

On the other hand, QuEra Computing's flagship system, Aquila, is a 256-qubit field-programmable qubit array (FPQA) operating as an analog Hamiltonian simulator. Using reconfigurable 2D arrays of rubidium atoms and Rydberg-mediated interactions, Aquila performs programmable coherent dynamics for applications ranging from quantum simulation to combinatorial optimization, including Maximum Independent Set problems. The platform is accessible via Amazon Braket, allowing users to design and execute pulse-level analog programs through QuEra's open-source Bloqade Python SDK.

Building on this foundation, Gemini, QuEra's next-generation processor, is designed for dual analog-digital operation and integrates mid-circuit measurement, error mitigation, and feedback control, marking key steps toward fault-tolerant architectures. QuEra ultimately aims to combine digital gate control with scalable logical qubit encoding, enabling the transition from today's NISQ devices to large-scale, universal quantum computers [51].

3 Problem Formulation and Embedding

3.1 The Problem

In emergency scenarios, establishing a reliable communication network is critical to ensure effective coordination among rescuers and control centers. To achieve this, we aim to deploy and optimise a network of antennas that can provide signal coverage to devices distributed across a target region. However, due to limited available resources, only a specific number of antennas can be activated simultaneously. For efficient communication, the active antennas should collectively maximise coverage while minimising overlap between neighbouring signals, thereby reducing overall power consumption and operational costs. Determining the optimal configuration of active antennas under these constraints constitutes a discrete optimisation problem, which is typically NP-hard (Def. B.12).

To formulate the problem, consider N possible sites, labeled $i = 0, \dots, N-1$, each associated with a binary variable $x_i \in \{0, 1\}$ indicating the presence (1) or absence (0) of an antenna. The antennas are to be placed within a region populated by devices, with N_t antennas available in total. Each antenna i covers a region B_i containing $N_d(B_i)$ devices. Our goal is to place the antennas so as to maximize the total number of covered devices while avoiding overcounting in overlapping regions. The **cost function** thus reads:

$$J(x) = \sum_i N_d(B_i)x_i - \sum_{i<j} N_d(B_i \cap B_j)x_i x_j + \sum_{i<j<k} N_d(B_i \cap B_j \cap B_k)x_i x_j x_k \tag{3.1}$$

where terms up to third-order overlaps are included. The total number of available antennas is limited by the following *inequality constraint*[1]:

$$\sum_i x_i \leq N_t \tag{3.2}$$

In this work, we consider a simpler yet analogous version of the problem. Instead of maximizing the number of covered devices, we maximize the total area covered by omnidirectional antennas, following [94]. Given N sites labeled $i = 1, 2, \dots,$ each antenna covers a circular region B_i with area $\mathcal{A}_i = \text{area}(B_i)$, pairwise overlaps $\mathcal{A}_{ij} = \text{area}(B_i \cap B_j)$, and triple overlaps $\mathcal{A}_{ijk} = \text{area}(B_i \cap B_j \cap B_k)$. The cost function becomes:

$$H_c(x) = -\sum_i \mathcal{A}_i x_i + \sum_{i<j} \mathcal{A}_{ij} x_i x_j - \sum_{i<j<k} \mathcal{A}_{ijk} x_i x_j x_k \tag{3.3}$$

where the overall negative sign converts the task into a *minimization* problem. This formulation corresponds to the limiting case of a homogeneous and densely distributed set of devices. The binary variables x_i now indicate whether each antenna is active, and the goal is to maximize coverage while minimizing overlap.

[1] As previously discussed, such constraints can be addressed through the introduction of slack variables.

For example, Fig. 3.1 shows the map of Sicily with possible antenna locations (black dots) of arbitrary coverage radius[2]. Active and inactive antennas are marked in green and blue, respectively. In the corresponding graph, each node represents an antenna i with a weight, and each edge corresponds to a non-zero entry of the connectivity matrix J_{ij} [94].

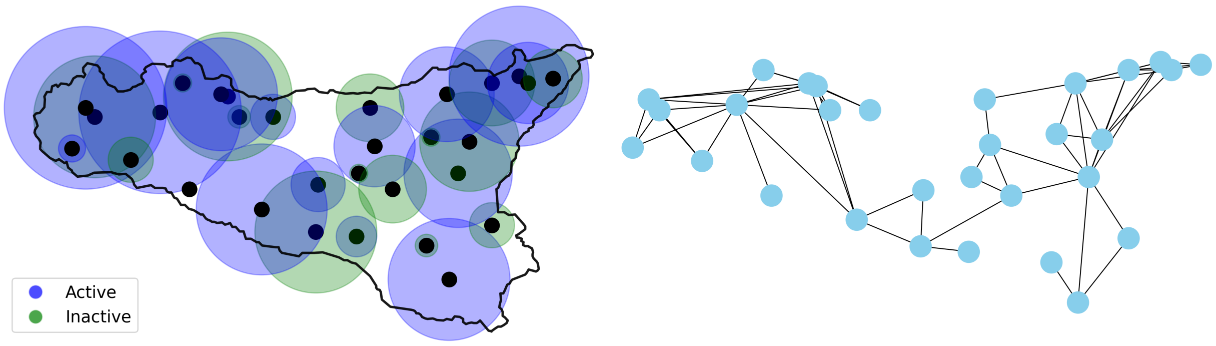


Figure 3.1: *Optimal solution of a selected antenna problem with N = 30 sites (left) and corresponding graph describing the connectivity of the considered problem (right)* [94].

We can enforce the number constraint (3.2) by introducing a **Lagrange multiplier** [95], denoted by p_n, into the cost function (3.3). The value of this multiplier can be adjusted to control the relative strength of the constraint term.

$$H(x) = H_c(x) + p_n H_n(x) \tag{3.4}$$

$$H_n(x) = \left(\sum_i x_i - N_t \right)^2 \tag{3.5}$$

To tackle this problem, we reformulate Eq. (3.4) into the QUBO form (1.12) by converting the cubic terms in Eq. (3.3) into quadratic ones. This is done using the **pairwise degree reduction** technique [96], which replaces each cubic term $x_i x_j x_k$ with a slack variable z_{ijk}. For each non-zero triplet (i, j, k), the substitution is:

$$x_i x_j x_k \;\longrightarrow\; x_i(1 - z_{ijk}) + \lambda_{ijk} M(x_j, x_k, z_{ijk}) \tag{3.6}$$

where λ_{ijk} is a *Lagrange multiplier*, and M is a **penalty function** that is *negative semi-definite* and vanishes if and only if $z_{ijk} = 1 - x_j x_k$[3] (i.e the slack variable is the complement of the product $x_i x_j$). This penalty term is added to the total cost function as

$$M(x_i, x_j, z) = -x_i x_j + 2(1 - z)(x_i + x_j) - 3(1 - z) \tag{3.7}$$

With this choice, the resulting QUBO matrix Q is real, symmetric, and has negative diagonal entries (linear terms $\mathcal{A}_i$) and positive off-diagonal ones (quadratic terms). Its size is $N + N_{\text{slack}}$, with N_{slack} the number of introduced slack variables, and it depends parametrically on the Lagrange multipliers λ_{ijk} and p_n.

[2] In this work, we will consider antennas with identical radii. This simplification eases representation and encoding, while remaining equivalent since the antenna weights account for differences in coverage.

[3] We work with the z in this form so that the coefficients of all quadratic terms are positive.

3.2 The Embedding Problem

As mentioned earlier, our goal is to model the cost function using a 2D array of Rydberg atoms driven by external laser pulses in the analog regime. We therefore reconsider the Rydberg Hamiltonian (2.9) and decompose it into two parts: the **mixing Hamiltonian** and the **cost Hamiltonian**, respectively given by

$$H_M(x) = \frac{\Omega(t)}{2}\sum_i \sigma_i^x \tag{3.8}$$

$$H_C(x) = -\Delta(t)\sum_i n_i + C_6\sum_{i<j}\frac{n_i n_j}{x_{ij}^6} \tag{3.9}$$

To implement the QUBO cost function using Rydberg atoms, we must map it onto the Hamiltonian H_C. This involves two main steps. First, we set the detunings such that $\Delta_i \sim -Q_{ii}$, which can be done directly by adjusting them experimentally.

Second, we arrange the atoms so that their interaction strengths satisfy $V_{ij} = Q_{ij}$ for $i < j$. In other words, realizing the QUBO with Rydberg atoms requires finding the optimal atomic positions on a two-dimensional plane so that the Rydberg interactions reproduce as closely as possible the matrix $\widetilde{Q}$[4], an optimization problem in itself.

The latter task is more challenging; to tackle it, we use a **principal component analysis** (**PCA**) technique [97] to determine the optimal planar embedding of the $\widetilde{Q}$ matrix elements. Obtaining the principal components requires a few algebraic steps. First, we invert the elements of Q_{ij} to construct a corresponding *distance matrix*, shifting the zero entries of Q by a small offset to avoid singularities:

$$R_{ij} = \sqrt[6]{\frac{C_6}{Q_{ij}}} \tag{3.10}$$

In particular, in this context

$$R_{ij} = \sqrt{(\mathbf{y}_i - \mathbf{y}_j)\cdot(\mathbf{y}_j - \mathbf{y}_i)} \tag{3.11}$$

where $\mathbf{y}_i$ denote the atomic coordinates. We set $\mathbf{y}_0 = (0,0)$ as the origin, and from the distance matrix (3.11) we can extract the **Gram matrix** [98] as follows:

$$M_{ij} = \mathbf{y}_i \cdot \mathbf{y}_j = \frac{1}{2}\left(R_{i1}^2 + R_{1j}^2 - R_{ij}^2\right) \tag{3.12}$$

To retrieve the point coordinates from M_{ij}, we compute its *SVD* (A.44):

$$M = U\sigma V^* \tag{3.13}$$

Up to an arbitrary rotation, the coordinates corresponding to the distance matrix R_{ij} are obtained as $Y = U\sqrt{\sigma}$, where the square root acts on the *eigenvalues* of the matrix.

[4] We will denote with this symbol the off-diagonal part of the QUBO matrix.

For atoms confined to a 2D plane, both M and Y are rank 2, so only the first two columns of Y contain non-zero coordinates. In the general case, M has rank ≥ 2, yielding points in an n-dimensional space. We then apply dimensionality reduction by retaining only the two largest singular values in the SVD of M. The resulting coordinates provide the planar configuration that best approximates the target matrix Q, as follows from the **Eckart-Young-Mirsky theorem** [99, 100].

This approach performs well for small problem sizes ($N = 3$ or $N = 4$) but quickly degrades for larger QUBO matrices, as the projection quality deteriorates. To understand the issue that arises when embedding the slack variables introduced during degree reduction, let us consider a simple toy model, illustrated in Fig. 3.2, consisting of three antennas placed at the vertices of an equilateral triangle with unit side length.

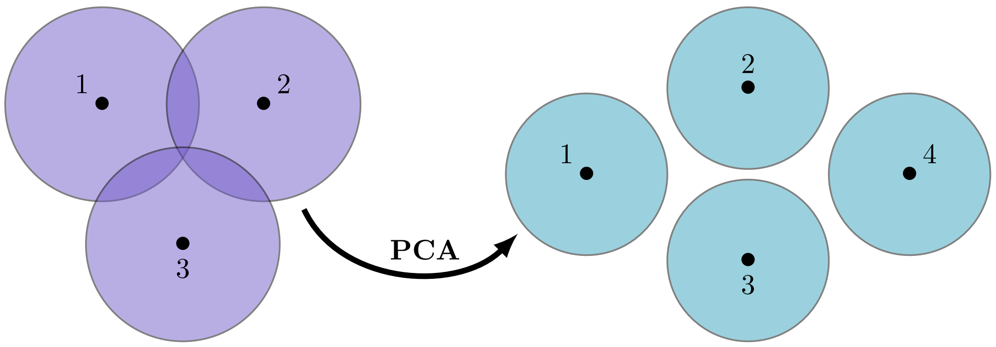


Figure 3.2: *Initial spatial configuration of antennas (purple) and resulting two-dimensional atomic register projection obtained through PCA (light blue), with the additional fourth atom representing the slack variable.*

Ideally, in the final QUBO matrix, the slack qubit should be equally and strongly coupled to the three original qubits, while remaining uncoupled from all other variables. By properly tuning[5] the Lagrange multiplier λ, the resulting register acquires a diamond-shaped configuration[6], as shown in Fig. 3.2.

From this simple example, a great limitation already emerges. In a more realistic setting, involving more than three antennas and a denser interaction graph, this approach can only succeed if the overlapping triangle lies along the edge of the larger structure. When the triangle is located within the interior, the slack qubit inevitably falls in proximity to other atoms, leading to unintended couplings and breaking the intended QUBO connectivity.

A natural way to address this issue is to lift the slack variables into a higher plane, effectively extending the problem to three dimensions. The additional degree of freedom introduced by the third axis allows us to embed the required couplings by tuning the "height" h at which each slack qubit is placed. The main drawback of this strategy, however, is that, as already mentioned, most Rydberg-atom quantum processors support only planar (2D) geometries. We therefore need a method to project the 3D configuration back onto a 2D plane while preserving the interaction topology, as will be discussed next.

[5] If λ is too small, the slack variable lies far from the plane, resulting in a poor projection.

[6] No Rydberg blockade is applied here (no hard constraints), with the classical optimizer determining the configuration. If atoms get too close, Ω must be tuned to adjust the blockade radius (2.4).

3.3 Baseline Embedding Model

Before detailing the algorithm used to solve the embedding problem, we outline its core idea, following [101]. In Rydberg-atom arrays, interatomic distances are bounded below by the Rayleigh range and above by the Rydberg blockade distance, which strongly constrain the geometry and topology of qubit couplings [68]. Consequently, when such arrays are used to solve combinatorial problems, such as finding the MIS of a given graph, a key limitation arises from the mismatch between distance-dependent atomic interactions and the distance-independent edges of the target graph, as already discussed.

One of the latest advances in Rydberg-atom technology to overcome this limit is **quantum wiring**, the use of auxiliary atoms to mediate interactions between otherwise distant atoms. This technique enables the realization of complex structures, including nonplanar and high-degree graphs[7], that would otherwise be unattainable [102,103]. *Quantum wires* can thus be exploited to perform topology-preserving transformations, effectively mapping a three-dimensional Rydberg-atom configuration onto a planar geometry.

We thus show here how three *Platonic graphs*[8] (tetrahedron, cube, and octahedron) can be experimentally implemented, and how their many-body ground states, corresponding to the MISs of these graphs, are probed through *quantum adiabatic control* techniques [81].

As shown in Fig. 3.3 (where the yellow edges are to be replaced by quantum wires), the Platonic solids are three-dimensional, yet their corresponding graphs are planar and can, in principle, be implemented on a plane. However, realizing these planar graphs with atoms arranged in two dimensions requires coupling atoms that are not physically adjacent. We also note that the employed quantum wires, are always composed of an even number of atoms, ensuring that the excitation alternation along the wire preserves the effective adjacency of the original atoms within the blockade regime.

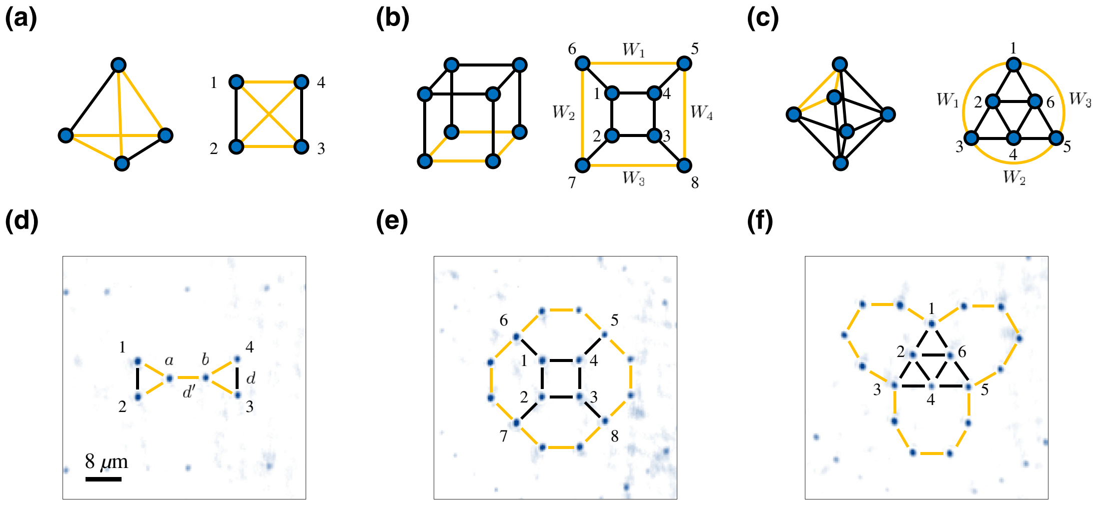


Figure 3.3: *The Platonic graphs and their two-dimensional arrangements* [101].

[7] A nonplanar graph cannot be drawn in such a way that no edges cross each other, while a high-degree graph is one in which the average number of edges per node (the degree) is large.

[8] Platonic graphs are the vertex-edge graphs of the five Platonic solids (tetrahedron, cube, octahedron, dodecahedron and icosahedron), i.e. convex and regular polyhedrons whose faces are congruent.

Formally, we model the atom array as a graph $G(V, E)$, where vertices V represent atoms and edges E represent pairwise interactions. In the Rydberg blockade regime, the system Hamiltonian H is given by Eq. (2.9). Selected edges of G can be replaced by quantum wires, either to reproduce the full Hamiltonian H using *local addressing fields*, or to reproduce only its ground-state spin configurations via *nonlocal addressing fields* and *postselective compilation*. Since the former remains experimentally challenging, in this work we adopt the latter version of quantum wires, with nonlocal addressing.

In the MIS phase, defined by $\Omega = 0$ and $0 < \Delta < U$ (with $U = C_6/x^6$) in Eq. (2.9), no adjacent atoms are simultaneously excited to the Rydberg state. Each edge of G can therefore be replaced by a quantum wire, whose antiferromagnetic (AF) configuration preserves the blockade condition without adding energy, if the wire's state is either $|00\rangle_W$, $|01\rangle_W$, or $|10\rangle_W$ [103]. The phase diagrams of $H(U, \Delta, \Omega = 0)$ for the Platonic graphs are shown in Fig. 3.4. Here, the detuning term favors Rydberg excitations (up spins), while the interaction term favors nonadjacent ones, giving rise to two paramagnetic phases, $\text{PM}_\downarrow$ for $\Delta < 0$ and $\text{PM}_\uparrow$ for $\Delta > 4U$, and intermediate, graph-dependent AF-like phases, among which the MIS phase is of primary interest.

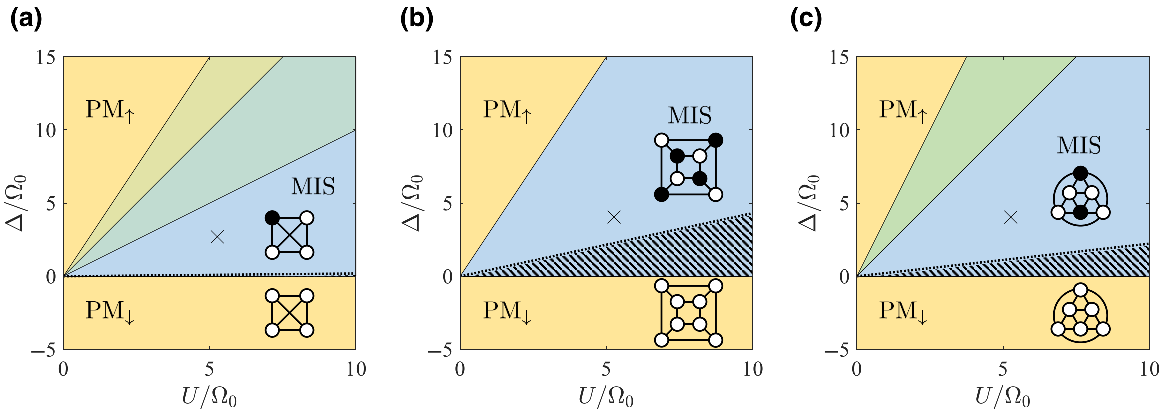


Figure 3.4: *Phase diagrams of the Platonic graphs. The diagrams show the ground-state spin configurations of* $H(U, \Delta, \Omega = 0)$*, highlighting the MIS phases (blue) and the shifts of phase boundaries due to neglected long-range interactions in Eq.* (2.9) [101].

This property is crucial for our purpose, as it enables (at least, in principle) the mapping of *any* three-dimensional graph onto a two-dimensional Rydberg array *without changing its ground state*. Consequently, the MIS of the original 3D graph can be faithfully obtained from its 2D embedding, providing a viable strategy for solving the target problem.

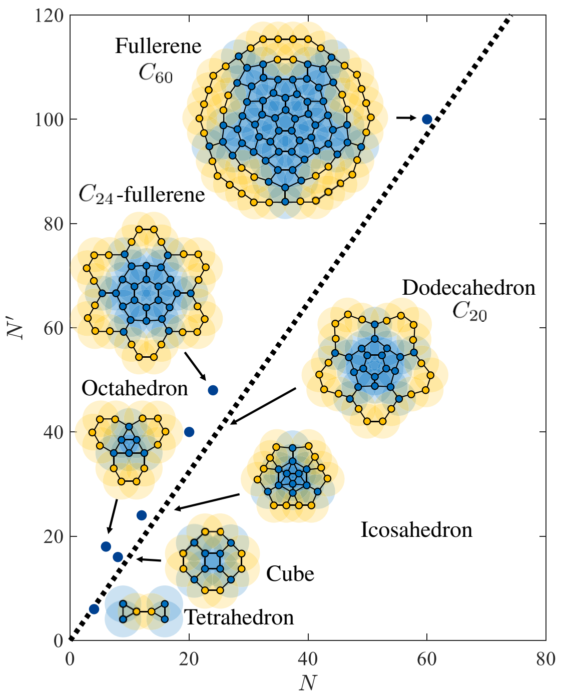
Moreover, as shown in [101], by tuning the wire and graph edge distances to $d' = d < d_R$[9], the system preserves equal coupling strengths while remaining in the blockade regime. Adiabatic driving from the paramagnetic to the antiferromagnetic phase reliably yields the MIS ground states of the Platonic graphs, with measured spin configurations matching theoretical and numerical predictions, when the AF wire condition holds. The results validate quantum wiring as a topology-preserving, scalable method for embedding nonplanar graphs and solving combinatorial problems using Rydberg-atom platforms.

[9] Here, d' denotes the interatomic distance between atoms within the wire, d the distance between connected nodes in the original (unwired) graph, and d_R the Rydberg blockade radius defined in Eq. (2.4).

In Fig. 3.5, we see the relation between N and N' for the three Platonic graphs discussed so far, together with additional examples including the remaining Platonic graphs (the icosahedron and dodecahedron) and two fullerene graphs, C_{24} and C_{60}. The data reveal an approximately linear scaling between N and N'.

Although this indicates a favorable scaling of atom number in the 3D-to-2D transformation, the achievable N' is in practice constrained by factors such as the *rearrangement probability*, which can be improved under cryogenic trapping conditions [104], and *bit-flip errors*, significantly reduced in *alkaline-earth atomic systems* [105]. More importantly, the scalability also depends on the *regularity of the target graph*: for more general, non-Platonic, or irregular 3D graphs, the scaling may deviate from linearity.

Finally, the expected *fidelity* with a specific MIS solution decreases as $\mathcal{O}(1/N)$, since roughly N equivalent AF ground-state configurations exist when quantum wires operate independently [101].



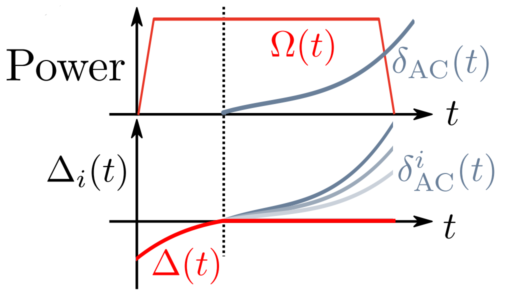


Figure 3.5: ***Left:*** *Scaling of the atom number N' in the quantum-wired graph G' as a function of the number of vertices N in the target graph G* [101]. ***Right:*** *In this protocol the global detuning Δ is swept from negative to positive values, while local light shifts δ_{AC} set the site-dependent detunings Δ_i encoding the graph weights.* [106]

Furthermore, a very recent advancement in the quantum-wire technique is the implementation of tunable *weights* directly in the wire atoms [106, 107]. This capability is achieved by introducing programmable local light shifts generated via **spatial light modulators** (**SLMs**) [108], which enable precise, site-dependent control of the atomic detunings.

The system is driven toward the ground state encoding the solution of the weighted graph through a *dual-stage adiabatic protocol* (Fig. 3.5). In this scheme, the local detunings are set as $\Delta_i = \Delta + w_i\delta_{\mathrm{AC}}$, where Δ is the global detuning and δ_{AC} the unit light-shift scale, constrained by $\max_i(\delta^i_{\mathrm{AC}}) < U(a)$ to preserve the Rydberg blockade at spacing a. The combination of global laser control and locally programmed light shifts enables smooth annealing into the target ground state [106].

In [107] this approach is shown to offer several major advantages. Since the wires mediate long-range interactions rather than acting as physical qubits, the required resources are significantly reduced.

The *spectral gap*[10] scales polynomially with system size, and the encoding remains robust to fluctuations in local detunings, allowing for faster adiabatic evolution and better-isolated ground states, which in turn enhance reliability and mitigate decoherence.

The architecture also withstands random variations in local light shifts with only a modest fidelity loss that increases with wire length; this can be further improved by fine-tuning the internal wire weights. Embedding the wires within larger graphs is additionally expected to enhance robustness through the extra energy gaps introduced by the logical structure and residual long-range van der Waals interactions.

The robustness of the *quantum wiring* technique, its capability to implement QUBO problems, and the ability to map 3D graphs onto topologically equivalent 2D graphs form the key motivations for the embedding algorithm, which we now analyze in detail.

3.4 The Embedding Algorithm

At the core of our algorithm lies the strategy illustrated in Fig. 3.3(a) and (d): each pyramid (while our graphs will not feature tetrahedra, they will include pyramids with triangular bases sufficiently regular to be compatible with this method) is remapped into a *bow-tie-like structure.* However, depending on how the instance parameters are chosen, more complex configurations than isolated pyramids may emerge, as we will discuss now.

To begin with, we must specify the class of instances that the algorithm is meant to address. First, the generated points are required to be generated within a fixed convex region, namely a regular but asymmetric hexagon. Then, in order not to impose too many constraints on our instances, we require them to satisfy only two main properties: *randomness* (with the possibility of reproducibility via seeds) and, more importantly, *uniformity*, so that even when the number of nodes is not sufficient to fully cover the region, they are still distributed evenly across it.

Once the points are generated, we construct the instance as follows: each point is assigned the same fixed radius, and an edge between two points is added when their Euclidean distance is at most this radius. In the special case where the radius is set to one, this yields a **unit disk graph** (**UDG**) [109]. In practice, we will vary the radius to study how the resulting instances, and consequently the performance of our algorithm, change. Nonetheless, the defining property of the UDG model effectively persists as long as all nodes share the same radius, since different choices correspond simply to rescaling distances.

The defining property of UDGs is precisely what motivates the requirement that points be uniformly distributed over the region: if the generated points start forming dense clusters, then even for small disk radii the resulting graph becomes highly connected, which makes it very difficult, and in practice almost impossible, to embed efficiently.

A first natural option for point's generation, widely used in simulation codes, is **Poisson disk sampling** (**PDS**), proposed by Cook in 1986 [110]. In PDS, points are placed randomly but subject to the constraint that no two points lie closer than a *prescribed minimum distance*, yielding a distribution that is both random and well-spaced. In particular, Bridson's 2007 algorithm [111] is one of the most widely used implementations of PDS, thanks to its high efficiency, good scaling properties, and simple, robust implementation.

[10] The spectral gap is the energy difference between the ground state and the first excited state of the system's Hamiltonian; a larger gap facilitates faster adiabatic evolution and more reliable solutions.

However, this approach has two main drawbacks. First, it introduces an additional simulation parameter: the minimum allowed distance between points. Second, Bridson's algorithm grows locally from its initial seed via an active front, so if sampling is stopped early (e.g. after a fixed small number of points), the resulting configuration may occupy only a local region around the seed, instead of covering uniformly the full area.

To avoid introducing additional simulation parameters, an alternative would be to start from a purely random sampling within the desired region and then apply **Lloyd's algorithm**, also known as **Voronoi iteration** or **relaxation**, which iteratively redistributes points to obtain evenly spaced configurations and partitions the domain into well-shaped, uniformly sized convex cells [112].

However, tests with a `python` implementation of the algorithm [113] revealed two issues. First, for some "unlucky" initial configurations, the relaxation only partially separated nearby points, still allowing highly clustered components. Second, and more importantly, the relaxation often pushed one or more points outside the target region, thereby violating the constraint on the required number of points inside the domain.

At this point, we would like to avoid introducing new parameters or moving points after their generation, so the sampling procedure itself should already produce a uniform distribution. In this context, the **Halton** [114] and **Sobol** sequences [115] are natural candidates. Both are deterministic *quasi-random* sequences, designed to mimic randomness while providing much more uniform coverage than purely random sampling. More precisely, they are *low-discrepancy sequences*, where the *discrepancy* quantifies how far the point distribution deviates from a perfectly uniform one; lower discrepancy therefore means that the points are spread more evenly across the domain [116].

Both Halton and Sobol sequences generate points in $[0, 1]^d$ by mapping successive integers to coordinates via specially structured number systems. In the Halton sequence, each coordinate is obtained by writing the index in a different *prime base* (e.g., base 2, 3, 5, ...), then reflecting its digits after the radix point (the so-called *radical-inverse function*). In the Sobol sequence, coordinates are built in base 2 using precomputed *direction numbers* (binary fractions with carefully chosen bit patterns) and combining them with the binary representation of the index through simple *bitwise operations* (such as XOR and shifts), which spread the points more uniformly across the domain.

A known drawback of the Halton sequence is that its *prime-base construction* can introduce visible correlation patterns and structured artefacts in the point set, so that the coverage of the domain is less isotropic than one would like. Various scrambled or permuted variants alleviate this, but the plain Halton sequence is often regarded as less robust than Sobol in practice. For this reason, we adopt the Sobol sequence, which typically yields a more uniform and visually pattern-free distribution in our setting.

Our overall construction is motivated by two considerations. First, the Sobol sequence combines deterministic low discrepancy with an easy randomisation scheme: by varying the scrambling seed we can generate diverse yet reproducible instances, and its regularity avoids point clustering while providing a uniform coverage of the domain, assuming the domain itself is sufficiently regular, as in our case. Second, UDGs arise naturally in neutral-atom quantum computing, where each node's radius corresponds directly to the physical blockade radius (2.4), making this geometric model an appropriate abstraction of the underlying hardware.

After the two-dimensional UDG is built, we impose constraints on possible third-order overlaps, i.e., configurations in which three disks of equal radius have a common intersection region. These constraints are encoded by introducing *slack variables* into the UDG, represented as additional nodes on a higher plane, each connected only to the three base nodes involved in the overlap. An example of such a 3D instance is shown in Fig. 3.6.

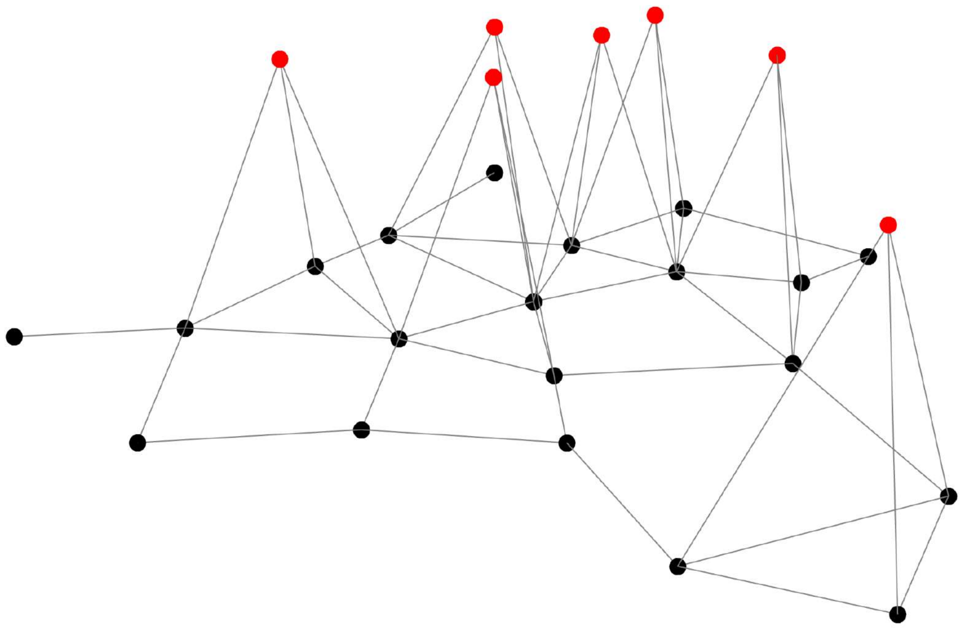


Figure 3.6: *3D UDG instance example visualised using the* `plotly` [117] *Python library. Red nodes represent* 3^{rd}*-order overlap's constraints.*

We introduce constraints on third-order overlaps for both graph-theoretical and physical reasons. From a combinatorial perspective, pairwise UDG edges cannot distinguish between three separate pairwise intersections and a genuine triple-overlap region. By adding slack variables, we explicitly encode this higher-order geometric feature and prevent degeneracies in the underlying structure.

From a physical standpoint, regions simultaneously covered by three blockade spheres give rise to complex and unintended three-body interactions that are not captured by pairwise proximity alone. Incorporating this constraint ensures that the resulting graph remains consistent with the actual geometry and operational limitations of the neutral-atom hardware, as well as with the structure of the MIS problem itself.

Having discussed how the instances are generated, we now turn to the local structures that can arise and that the algorithm must handle during the embedding process. In all the following figures (unless explicitly stated otherwise), the panel on the left shows the three-dimensional structures viewed from above: red nodes represent the *apex nodes* (i.e., the slack variables), while light-blue nodes represent the *base nodes* (i.e., the original points extracted from the Sobol sequence). The panel on the right shows the corresponding two-dimensional embedding of the same structure, now including the *wire nodes* in yellow.

The first natural extension of an isolated pyramid is a pair of pyramids sharing a single base node, as illustrated in Fig. 3.7. The embedding of this configuration is shown on the right of the same figure: the two bow-tie gadgets are attached at the shared base node, thereby preserving the minimal number of additional nodes (two wire nodes per pyramid) while ensuring that no *spurious edges* are introduced. Here, with "spurious edges" we specifically mean effective connections generated by an even number of wire nodes between vertices that were not adjacent in the original graph.

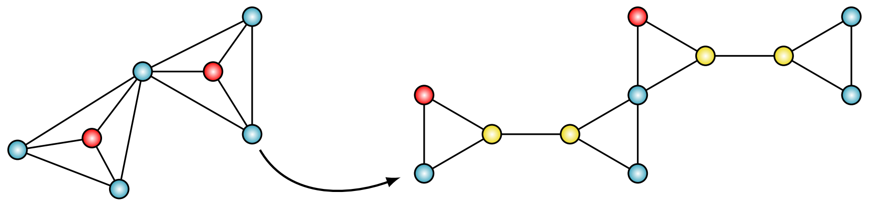


Figure 3.7: *Double-pyramid configuration with a shared base node and its embedding into two connected bow-tie structures.*

The next structure we consider is what we call a *star node*, arising when a single node is shared by three or more pyramids. In this section we distinguish between two types of star nodes. The first, simpler case is when the pyramids share *only* this one node and no other nodes, as shown in Fig. 3.8, where three pyramids meet at a single shared node.

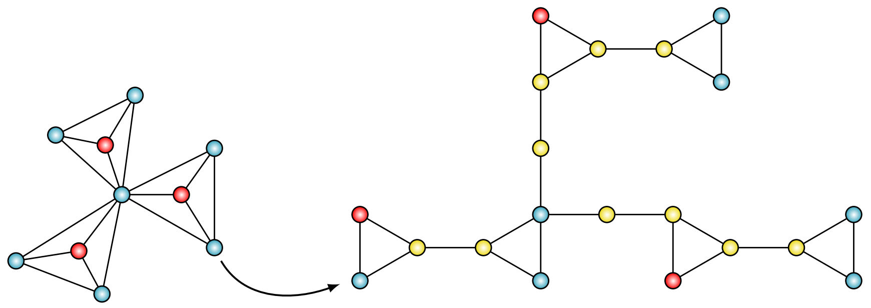


Figure 3.8: *Star-node configuration with three pyramids meeting at one base node and its corresponding 2D embedding.*

In the figure we explicitly show only the three-pyramid case, since configurations in which four or more pyramids share exactly one common node are quite rare in our instances. Nevertheless, the embedding algorithm can handle such higher-order star nodes in essentially the same way as the configuration depicted in Fig. 3.8.

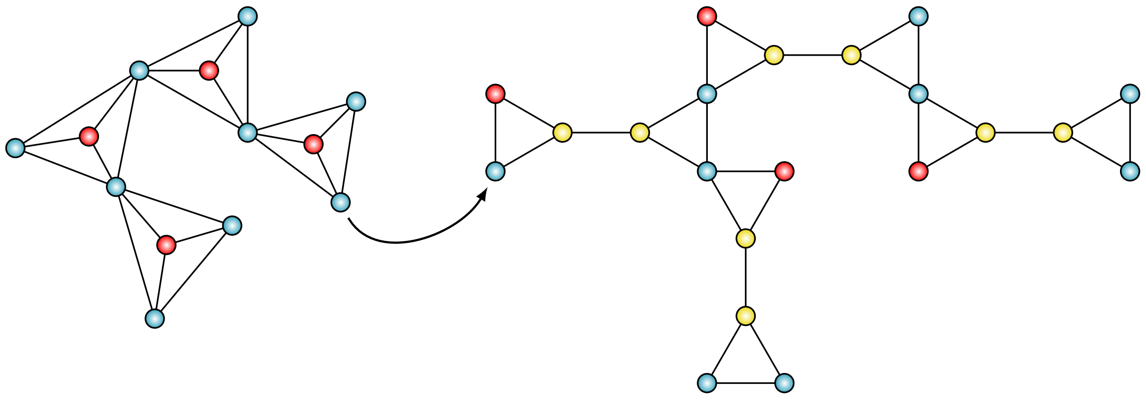


Figure 3.9: *Tree-like configuration of single-sharing pyramids and its 2D embedding.*

Combining the embedding rules illustrated in Fig. 3.7 and Fig. 3.8, we can also embed arbitrarily large configurations of pyramids, which we refer to as *trees*. Inside these *tree-like* structures, any two neighbouring pyramids share exactly one base node, and every pyramid is connected either directly through such a shared node or indirectly via a *branch* of pairwise single-sharing pyramids.

The embedding of this *single-sharing tree* configuration is shown in Fig. 3.9, where we also see that some pyramids are rotated with respect to the basic pattern in Fig. 3.7. This controlled rotation is introduced to prevent unwanted overlaps between pyramids that are not connected in the original configuration.

With these three embedding strategies, we can embed any structure composed exclusively of single-sharing pyramids. We now turn our attention to more complex configurations, formed by pyramids that share *two base nodes* or, equivalently, share a common *edge*.

The first and simplest structure of this kind is the natural extension of Fig. 3.7, corresponding to a pair of pyramids that share a single edge. As shown in Fig. 3.10, this configuration is embedded by placing the two shared nodes on the same side of the bow-tie, so that the endpoint of the first bow-tie coincides with the starting point of the second. This yields a compact embedding that uses the minimal number of additional wire nodes.

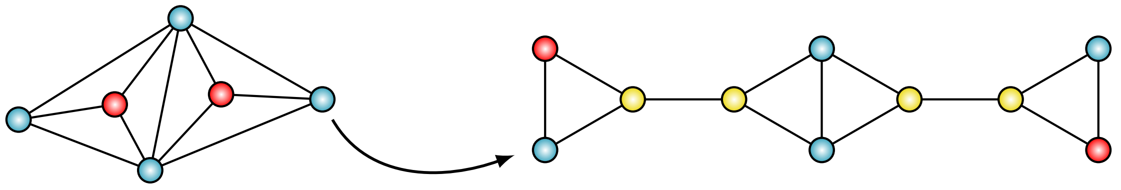


Figure 3.10: *Edge-sharing double-pyramid configuration and its bow-tie embedding.*

Now, the case in which *three* pyramids share the same edge pairwise is more interesting. As shown in Fig. 3.11, the first two pyramids are embedded exactly as in Fig. 3.10. However, the third bow-tie cannot be placed directly on top of the one with which it shares the edge: doing so would create unwanted connections between the upper pair of wire nodes and the lower pair, forming a *square* of wire nodes.

Such a configuration must always be avoided, as it would effectively introduce new edges and thus alter the original graph connectivity. For this reason, the third bow-tie is shifted further away, at the cost of four additional wire nodes.

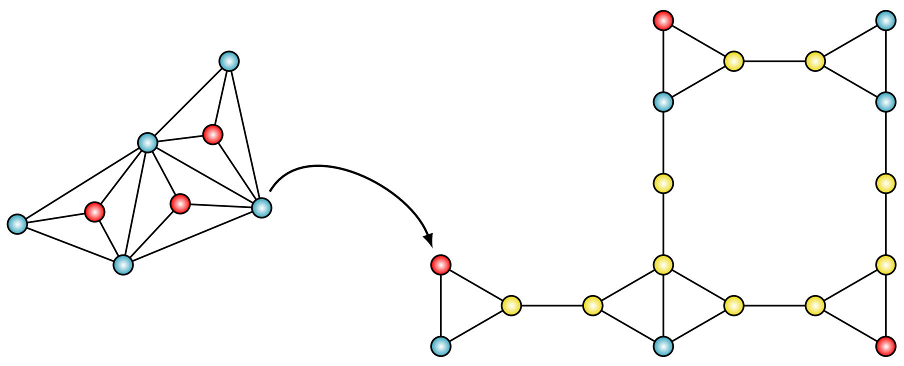


Figure 3.11: *Edge-sharing three-pyramid configuration and its corresponding triple bow-tie embedding avoiding wire squares.*

Things become interesting when we consider structures with more than three edge-sharing pyramids. We distinguish three types of such configurations: *clusters*, the second type of *star nodes*, and *chains*. Starting with the latter, an *edge-sharing chain* is defined as a sequence of pyramids where consecutive pyramids share exactly one edge pairwise, each pyramid shares *at most* two edges, and every base node belongs to *at most* three pyramids.

The embedding of such a chain proceeds by choosing one of the *extremal pyramids* (i.e., one that shares only a single edge) as the starting point and then iterating the pattern of Fig. 3.11: we alternately attach each new bow-tie on the right/left side of the previous one and then on its top/bottom side, thereby iteratively extending the chain arbitrarly while preserving the desired connectivity.

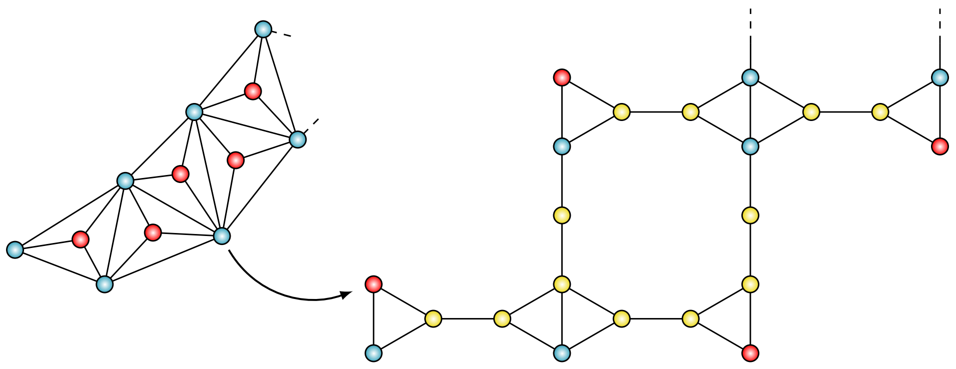


Figure 3.12: *Edge-sharing pyramid chain and its bow-tie embedding.*

Moving on to the *edge-sharing cluster*, we define it as the specific configuration shown in Fig. 3.13: a central pyramid shares each of its three base edges with a distinct neighbouring pyramid, and these three outer pyramids do not share edges among themselves. Any additional edge-sharing pyramid attached to this configuration would break the cluster structure and is therefore treated as a different type of configuration.

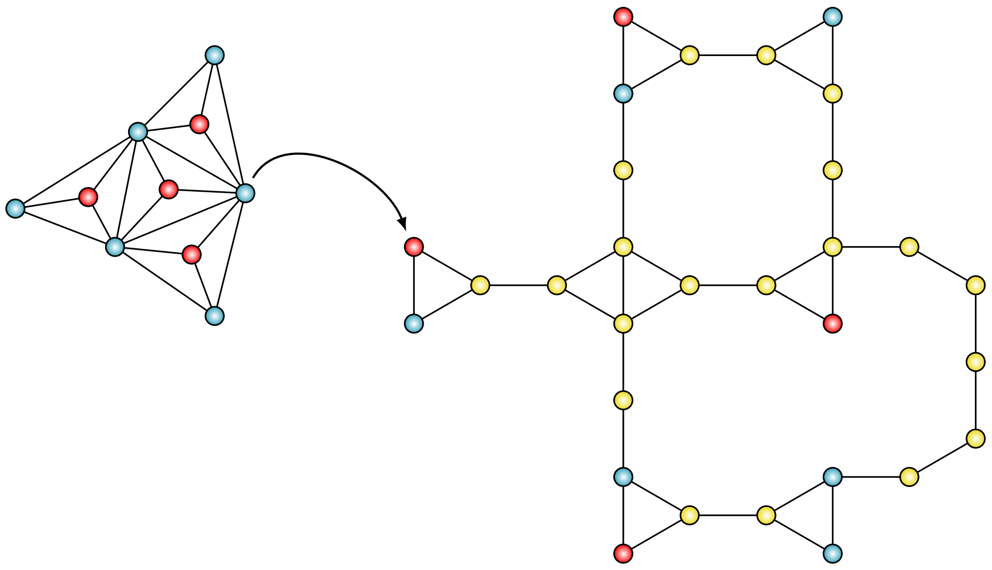


Figure 3.13: *Edge-sharing cluster with a central pyramid and three neighbouring pyramids, and its corresponding bow-tie embedding.*

The last structure we consider is the edge-sharing variant of the *star node*. This configuration is similar in spirit to the chain: we again have a sequence of pyramids, each sharing at most two edges. The key difference is that now every pyramid must include a common central *star node*, highlighted in purple in Fig. 3.14, while all other base nodes can be shared by at most two pyramids.

As shown in Fig. 3.14, after an initial bow-tie embedding analogous to Fig. 3.11, the construction proceeds "vertically": each new bow-tie is placed above the previous one. At each step, a specific wire pattern is added on the left to reconnect the star node back to the previous bow-tie, ensuring that this node remains connected to every pyramid as in in the original three-dimensional configuration.

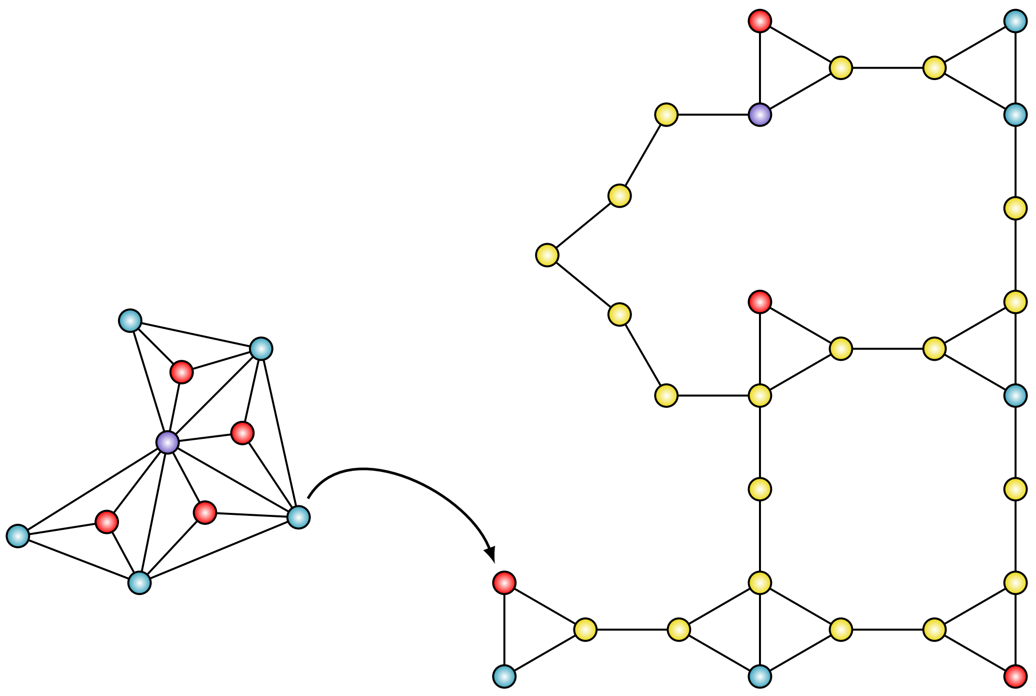


Figure 3.14: *Edge-sharing star-node configuration and its bow-tie embedding.*

Before describing the operation of the algorithm, it is useful to briefly explain how *loops* are handled. Intuitively, we call a loop any configuration of pyramids (either single-sharing or edge-sharing) in which, starting from a given pyramid and following shared nodes or edges, one eventually returns to the same pyramid. In our instances this may occur mainly in two situations: within a single-sharing tree (Fig. 3.9) and in the "closed" version of the edge-sharing star node, where all pyramids share two edges (Fig. 3.14).

In both cases loops are treated in the same way. During the embedding process, the algorithm keeps track of any node whose position changes as a consequence of placing a new bow-tie. Whenever a node is displaced, a path-finding routine is invoked to reconnect its new position to its original one through a quantum wire. This wire is constructed so that the distance between consecutive wire nodes is constant, the total number of wire nodes is even, and no wire node becomes connected to external nodes other than the designated start and end nodes.

We can now finally explain how the embedding algorithm operates:

1. First, an instance is generated from scratch, either using a provided seed or a random one. Points are sampled from the Sobol sequence under two constraints: they must lie inside the target region, and sampling continues until the desired number of nodes has been placed. Next, a two-dimensional UDG is built using the chosen radius, and third-order overlaps are then detected to add the slack nodes, yielding the three-dimensional starting instance.

2. Then, in its first phase, the algorithm iteratively searches for structures to embed, alternating between detection routines and the corresponding embedding functions. This loop terminates either when all pyramids have been successfully embedded or when a preset iteration threshold is reached. In the latter case, the algorithm returns a `False` flag to indicate that some pyramids, typically belonging to highly connected or particularly complex configurations not covered by our rules, have not been embedded. During this phase, the algorithm also builds a list of position dictionaries, each corresponding to the embedding of a specific structure.

3. If all pyramids are embedded successfully, the algorithm identifies the largest connected component and selects it as the starting point for the final layout. Since each component is initially constructed around the origin, rigid translations are then applied to position the components in the plane without overlaps.

4. Then, once the first component has been selected and the final layout is ready to be assembled, the algorithm enters a loop in which it performs the following checks:

   (a) At the beginning of each iteration, the algorithm checks whether any edges are missing in the current layout and, if so, restores them using the same path-finding routine employed for handling loops.

   (b) Then, it checks whether the new component shares a node with the current layout. If so, a *shift function* (Fig. 3.15) is used to "pull out" this node from the existing layout, and the new component is attached to it via a rigid translation. If this translation creates undesired edges, the algorithm then tries a sequence of *rotations* and *reflections* around the shared node, until it finds a placement that fits into the final layout without introducing spurious connections.

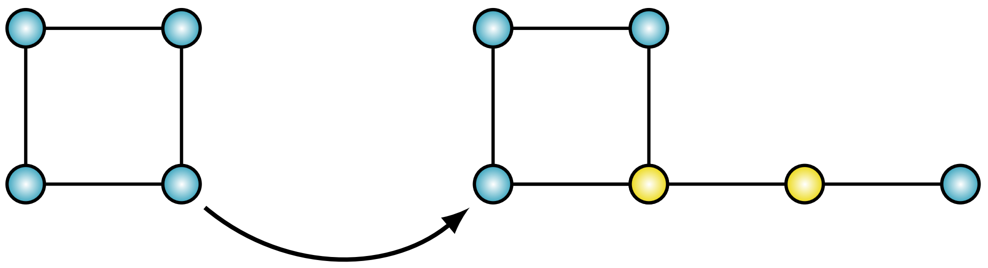


Figure 3.15: *Example usage of the shift function. Given a target node and a cardinal direction as input, the function displaces the node in that direction and restores its original connections by inserting two wire nodes.*

(c) Next, the algorithm checks whether the new component has any node that is adjacent (in the original graph) to a node already present in the layout. If this is the case, the same procedure is applied: the relevant node is shifted, the new component is translated to realise the desired connection, and, if needed, rotations and reflections are attempted until a valid placement is found.

(d) If no new components can be directly attached and no edges need to be restored, the algorithm then checks whether any non-pyramid node can be added to the layout. These nodes are not attached via the shift helper: instead, they are placed directly around their target node. If this direct placement fails, the algorithm connects the new node to a wire incident on the target node, ensuring that the number of intermediate wire nodes is even. After these placements, the routine again checks for missing edges.

(e) Finally, any remaining disconnected nodes or components, if present, are placed into the layout as well, positioned sufficiently far from the rest of the graph.

5. The loop terminates either when there are no remaining components or nodes to place, or when an error is raised because some of them cannot be placed.

6. At the end of the procedure, a dedicated *topology check* is performed to verify that all nodes and edges of the original 3D graph are correctly reproduced in the 2D configuration. If this check is passed, the function returns a dictionary containing all the new 2D positions, ready to be used for plotting.

Below, we give a schematic summary of the embedding algorithm, whose Python implementation can be found at [118].

Embedding Algorithm

Input: Seed, Number of Nodes
Output: 2D Positions of Nodes

```
1. Generate 3D instance
2. Detect and embed pyramids
3. Find largest component & prepare layout
4. Assemble final layout iterating over component and nodes:
   4a. Restore missing edges if present
   4b. Attach components sharing a node with the current layout
   4c. Attach components containing a node adjacent
       to a one present in the current layout
   4d. If no edges need restoring, attach non-pyramid
       nodes adjacent to nodes in current layout
   4e. Place disconnected nodes/components far from the main layout
5. Loop ends when all is placed
6. Run a topology check and return 2D positions if passed
```

4 Solving The Problem

4.1 The QAOA

Within the **noisy intermediate-scale quantum** (**NISQ**) era [119], **variational quantum algorithms** (**VQAs**) [120] have emerged as a promising approach to exploit the capabilities of current quantum hardware through a *hybrid quantum-classical optimization scheme*. In such algorithms, a parameterized quantum circuit is executed on a quantum processor, while a classical optimizer iteratively updates the circuit parameters by minimizing a cost function defined from the measurement outcomes of the quantum circuit. This hybrid structure enables VQAs to operate with relatively shallow circuits, thereby mitigating the detrimental effects of noise inherent in NISQ devices.

In particular, the **quantum approximate optimization algorithm** (**QAOA**) [10] is one of the most promising VQAs that has attracted great interest in recent years. QAOA is designed to find approximate solutions to hard combinatorial optimization problems on quantum computers: it encodes the Hamiltonian related to the problem into a quantum circuit and leverages adiabatic time evolution and layering to optimize the variational parameters of the circuit, such that the approximate solution to the problem can be constructed by measuring the QAOA circuit with the optimal set of parameters.

The fundamental building block, a single layer of the QAOA circuit (Fig. 4.1), consists of a **cost layer** associated with the problem Hamiltonian H_C and a **mixer layer** whose corresponding Hamiltonian H_M *does not commute*[1] with the problem Hamiltonian.

The performance of QAOA is typically measured by the approximation ratio $C_{\mathrm{QAOA}}/C_{\max}$ i.e., the ratio of the cost associated with the solution output by QAOA to that of the true optimal solution. Theoretically, such an approximation ratio increases with increasing layers p, as QAOA recovers the adiabatic evolution in the $p \to \infty$ limit [121].

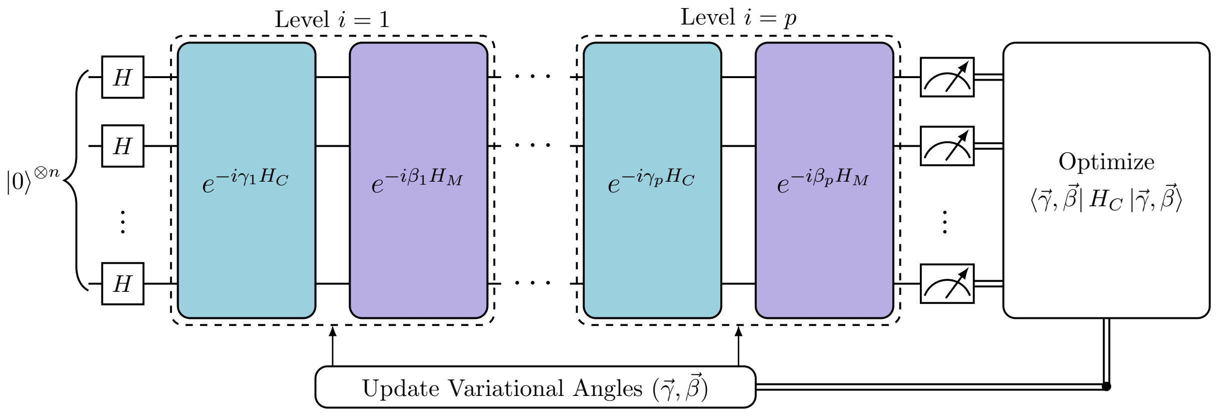


Figure 4.1: *Schematic representation of a QAOA circuit.*

[1] This property is fundamental for enabling the QAOA to explore the Hilbert space of the solution. If the cost Hamiltonian H_C and the mixing Hamiltonian H_M commuted, the evolution would be restricted to the initial subspace, preventing exploration of the broader solution space.

As already noted, QAOA is part of the broader class of VQAs, named after the **variational principle** [122]. This principle is used to find the lowest expectation value, typically the ground state energy, for a given observable. It relies on a trial wave function parameterized by a set of values that can be adjusted to fit the system's wave function and minimize the expectation value. Mathematically[2], this is expressed with a Hamiltonian H and trial wave function $|\psi\rangle$ to find the ground state energy E_0, bounded by:

$$E_0 \leq \frac{\langle\psi| H |\psi\rangle}{\langle\psi|\psi\rangle} \tag{4.1}$$

The goal of variational algorithms is to find a parametrization of $|\psi\rangle$ that minimizes the Hamiltonian's expectation value. This is done by iteratively updating the *parameters* of a fixed *ansatz*, an initial trial wave function whose functional form is chosen in advance based on the system's context. The choice of ansatz is tailored to the specific problem and to the structure of the system's Hamiltonian.

This principle is highly relevant in quantum computing because it directly applies to quantum systems. Qubits, as fundamental and versatile representations of wave functions, are measured to obtain the system's expectation value and energy. This process occurs at the end of the quantum circuit, after a series of *parameterized quantum gates* have modified the system and its wave function [121].

The first version of the QAOA [10], was inspired by the *Trotterized version* of the **quantum adiabatic algorithm (QAA)** [78,124], namely the evolution operator $U(t)$ describing the evolution of the QAA, is decomposed into a sequence of small steps[3] through the *Trotter-Suzuki formula* [125,126], which in this context can be recasted as follows [121]:

$$\begin{aligned} U(t) &\approx \prod_{k=0}^{r-1} \exp\Big[-iH\big(k\Delta\tau\big)\Delta\tau\Big] \\ &= \prod_{k=0}^{r-1} \exp\Big[-if\big(k\Delta\tau\big)H_C\Delta\tau\Big] \exp\Big[-ig\big(k\Delta\tau\big)H_M\Delta\tau\Big] \end{aligned} \tag{4.2}$$

where f and g are two (slowly varying) predefined control functions, $\Delta\tau \equiv t/r$, with $t \in [0, T]$ and T is the total evolution time. In the QAOA, the predetermined functions f and g are replaced by variational parameters γ_k and β_k, which are optimized during training. Specifically, the original QAOA[4] proceeds through the following steps [121]:

- Define a cost Hamiltonian H_C such that its lowest energy state encodes the solution to the optimization problem. Define also a mixer Hamiltonian H_M that does not commute with H_C.
- Initialize the circuit in the state $|s\rangle = |+\rangle^{\otimes n}$, where n is the number of qubits and $|+\rangle$ is the superposition state defined in Eq. (A.61).

[2] This form of the variational principle is known as the *Rayleigh-Ritz method* [123].

[3] In principle, the QAA involves a *continuous* evolution of the quantum state. To emulate this process on a gate-based quantum computer, the evolution must therefore be discretized.

[4] In the original formulation, QAOA is defined as a maximization problem; however, here we express it in the minimization form for consistency.

- Construct the circuit ansatz by defining and applying the unitaries:

$$\begin{aligned} U_C(\gamma) &= e^{-i\gamma H_C} \\ U_M(\beta) &= e^{-i\beta H_M} \end{aligned} \tag{4.3}$$

  where the parameters γ and β serve as the *variational parameters* of the circuit. We denote $U_C(\gamma)$ and $U_M(\beta)$ as the cost and mixer layers, respectively. A single QAOA layer consists of one cost layer followed by one mixer layer, and multiple such layers can be stacked to construct a deeper circuit.

- Define the total number of QAOA layers, $p \geq 1$. Initialize the $2p$ variational parameters $\vec{\gamma} = (\gamma_1, \gamma_2, \ldots, \gamma_p)$ and $\vec{\beta} = (\beta_1, \beta_2, \ldots, \beta_p)$ such that $\gamma_k \in [0, 2\pi)$ and $\beta_k \in [0, \pi)$ for $k = 1, \ldots, p$. The final state output by the circuit is given by

$$\left|\psi_p(\vec{\gamma}, \vec{\beta})\right\rangle = e^{-i\beta_p H_M} e^{-i\gamma_p H_C} \ldots e^{-i\beta_1 H_M} e^{-i\gamma_1 H_C} \left|s\right\rangle \tag{4.4}$$

- The expectation value of H_C with respect to the ansatz state (4.4), representing the cost evaluated by the quantum algorithm for the given problem, is obtained through repeated measurements of the final state in the computational basis:

$$F_p(\vec{\gamma}, \vec{\beta}) = \left\langle \psi_p(\vec{\gamma}, \vec{\beta}) \right| H_C \left| \psi_p(\vec{\gamma}, \vec{\beta}) \right\rangle \tag{4.5}$$

- A classical optimization algorithm is employed to iteratively update the parameters $\vec{\gamma}$ and $\vec{\beta}$. The goal of the aforementioned routine is to find the optimal set of parameters $(\vec{\gamma}^*, \vec{\beta}^*)$ such that the expectation value (4.5) is minimized.

This procedure is illustrated in Fig. 4.1. For a comprehensive overview of QAOA and its extensions, including alternative ansatz designs, classical optimization strategies, computational efficiency across problem classes, solution quality, noise sources and mitigation techniques, hardware-specific implementations, performance enhancements, experimental results, and prospective research directions, see [121].

4.2 The QIRO Algorithm

With the current state of the art, classical simulation of QAOA is severely limited by the 2^N scaling of the state-vector dimension. For generic random graphs without particular symmetry, state-of-the-art simulators typically reach at most $N \sim 30$ qubits. This represents a serious bottleneck for our problem, since after the embedding step, instances with larger N and higher interaction radii can easily reach sizes of order $N \sim 100$. Moreover, as the system size grows, the number of layers p required for QAOA to converge to a good solution generally increases, further increasing the cost of classical simulation.

For this reason, we choose to address the MIS problem on the final 2D graph using the **Quantum-Informed Recursive Optimization Algorithm** (**QIRO**) [12]. In QIRO, information generated by quantum resources is exploited to recursively shrink the optimization problem, guided by problem-specific classical optimization routines, as illustrated schematically in Fig. 4.2.

Another reason for this choice is the strong structural similarity between our graphs after the embedding algorithm and the instances QIRO was designed for. In both cases, the graphs are tailored for implementation on neutral-atom quantum computers, featuring short-range connectivity with only nearest-neighbour interactions.

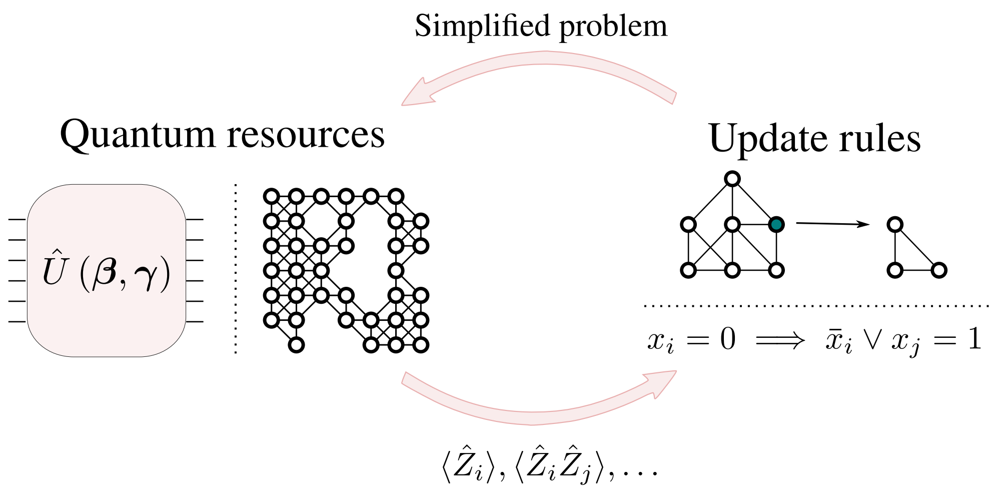

Simplified problem
Quantum resources
$\hat{U}(\boldsymbol{\beta}, \boldsymbol{\gamma})$
Update rules
$x_i = 0 \implies \bar{x}_i \vee x_j = 1$
$\langle \hat{Z}_i \rangle, \langle \hat{Z}_i \hat{Z}_j \rangle, \ldots$

Figure 4.2: *Scheme of the QIRO algorithm's core principles. Quantum resources (QAOA) are used to obtain information which is then used to simplify the problem through problem-specific classical update rules.* [12]

The motivation for augmenting QAOA with classical post-processing stems from its inherently *local* behavior, particularly at low depth and for small parameter values. In a depth-p circuit, each QAOA layer acts only on qubits connected by edges of the interaction graph, implying that the state of any qubit can depend only on qubits within graph distance $\mathcal{O}(p)$. Consequently, correlations cannot propagate far across the system: they remain confined within a finite *correlation length* determined by the circuit depth. This restricted locality limits the ability of shallow QAOA to capture global structure and can significantly hinder its performance on large problem instances [127, 128].

Since implementing genuinely *non-local* updates is hardware-expensive at the quantum level, recursive QAOA (RQAOA) [129, 130] introduces them classically: variable values are iteratively fixed by rounding QAOA-estimated correlations, reducing the problem size and effectively shortening distances in the interaction graph. This both mitigates QAOA's intrinsic locality and enables efficient classical simulation, as low-depth QAOA is relatively cheap to simulate. A Python implementation of QIRO is available at [131].

In [12] the authors present an implementation of QIRO for two NP-hard combinatorial optimization problems: MIS and *Maximum Satisfiability* (*MaxSAT*) [7], the optimization counterpart of *SAT*, the first problem proven to be NP-complete (Def. B.12). Since our application targets MIS, we will restrict the following discussion to this case.

Since QIRO relies on problem-specific update rules, the relevant optimization problems are first formally mapped to quantum Hamiltonians. For a binary vector $x \in 0, 1^*$, finding a MIS is equivalent to finding the ground state of the following classical Hamiltonian:

$$H(x) = -\sum_i x_i + \lambda \sum_{ij} x_i x_j \tag{4.6}$$

where $\lambda > 1$ is a penalty factor to enforce the independence constraint [36].

The Hamiltonian (4.6) is completely general: it does not rely on any particular structure of the interaction graph beyond minimal assumptions, namely, that the graph is *undirected* (edges have no orientation; interactions are symmetric) and contains *no self-loops* (no node interacts with itself). These minimal constraints suffice for the Hamiltonian to correctly encode the MIS problem for arbitrary graphs.

To go then from the classical Hamiltonian (4.6) to the quantum one one performs the mapping $s_i = 2x_i - 1$ to Ising variables $s_1 = \pm 1$, and then promoting the spin variables to quantum operators $s_i \to \sigma_i^z$ we obtain the quantum version of (1.15).

The first step of QIRO is *quantum state preparation*. At each iteration, the goal is to prepare a state with low expectation value with respect to the cost Hamiltonian H_C, so that the classical reduction is guided by (superpositions of) configurations corresponding to good candidate solutions. To generate such low-energy states, QAOA is run at low depth ($p \leq 3$) using the following mixing and cost Hamiltonians:

$$H_C = \sum_i h_i \sigma_i^z + \sum_{ij} J_{ij} \sigma_i^z \sigma_J^z \tag{4.7}$$

$$H_M = -\sum_i \sigma_i^x \tag{4.8}$$

The $2p$ parameters $(\boldsymbol{\beta}, \boldsymbol{\gamma})$ are then determined by a classical optimization routine such that (4.5) is minimized, as already described in Sec. 4.1.

After this step, one- and two-point correlators, $\langle\psi| \sigma_i^z |\psi\rangle$ and $\langle\psi| \sigma_i^z \sigma_j^z |\psi\rangle$, are computed and stored in a matrix M. In an ideal numerical simulation, only entries M_{ij} corresponding to edges of the original interaction graph are nonzero, so the update rules are designed to use correlations only between such connected variables.

Next, the update step selects[5] the largest entry of M in terms of its absolute value; if several entries share this maximal value, one of them is chosen at random. It then checks whether the selected entry is diagonal (corresponding to a one-point correlation) or off-diagonal (corresponding to a two-point correlation)
This step yields four possible update actions (Fig. 4.3):

a) If $M_{ii} \geq 0$ was selected, the i-th vertex is set to be in the independent set (IS). Then, all vertices connected to the i-th node are removed from the graph, as including them would violate the independence constraint.

b) If $M_{ii} < 0$ is selected, the i-th vertex is removed from the graph.

c) If $M_{ij} > 0$ is selected, both nodes are removed from the graph, because the only positively correlated assignment of connected vertices consistent with the independence constraint is not to include either node in the IS.

d) If $M_{ij} < 0$ is selected, all vertices adjacent to both i and j are removed. Intuitively, a negative correlation suggests that either i or j will be in the IS, so any vertex connected to both simultaneously cannot be included.

[5] Intuitively, the largest correlation identifies a variable (or pair) that appears with high probability in the low-energy state $|\psi\rangle$, making it a natural candidate to guide the update step.

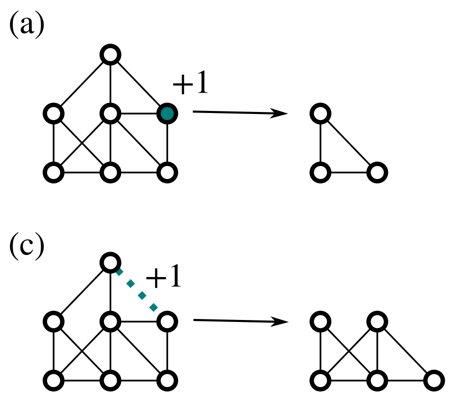


Figure 4.3: *Practical example of the MIS update rules described.* [12]

If no nodes were removed from the graph using these simplifications, the procedure is repeated using the next largest correlation in terms of its absolute value. In addition, if any connected component with $n_c \leq 15$ vertices is found, it is solved exactly by brute-force search, i.e. by enumerating all possible spin configurations and selecting the one with lowest energy, for its MIS and then removed from the main graph. This step is mainly intended to save resources when running the algorithm on real quantum hardware.

The whole procedure is repeated until the size of the problem is reduced sufficiently, to be able to solve it exactly by a brute-force search or other means if the reduction does not yield problem sizes amenable to exact solvers.

The update rules are tailored to the MIS problem and explicitly enforce the independence constraint. Unlike the RQAOA rules, the QIRO MIS updates are local in the previously defined sense, as they never create new couplings between variables. However, in RQAOA, the intermediate Hamiltonians generically leave the MIS form, whereas in QIRO each reduced instance remains a valid MIS problem. Consequently, any analog encoding of the original MIS instance can be reused for all intermediate, simplified problems.

As mentioned, one of the main reasons for selecting QIRO for our tests is the strong similarity between the QIRO instances and the final graphs obtained after applying the embedding algorithm. Specifically, QIRO has been benchmarked UDGs on square lattices with sizes in the range $n \in [40, 200]$, using 50 randomly generated UDGs for each n. An example of a graph instance with $n = 137$ nodes is shown in Fig. 4.4, where the teal nodes indicate those belonging to the MIS, which contains 45 nodes in this instance. For comparison, one configuration obtained through the algorithm is shown in Fig. C.3.

Among the results reported in [12], it is worth mentioning that QIRO produces high-quality feasible solutions even for *suboptimal parameter choices*, highlighting the robustness of the algorithm against lower-quality quantum information.

Moreover, experiments on the Aquila quantum device [51] show that stronger quantum correlations enhance QIRO's performance (Fig. 4.5). This is because analog quantum devices naturally generate non-local quantum correlations through many-body dynamics, which are classically hard to simulate, especially for $p > 1$.

In Fig. 4.5, the *approximation ratio* is defined as the ratio between the *cardinality* (i.e., the number of nodes) of the independent set found by QIRO and the size of the MIS obtained via the tensor network algorithm from [132], used as classical benchmark to test QIRO's performance, due to its high efficiency in tackling this type of problems.

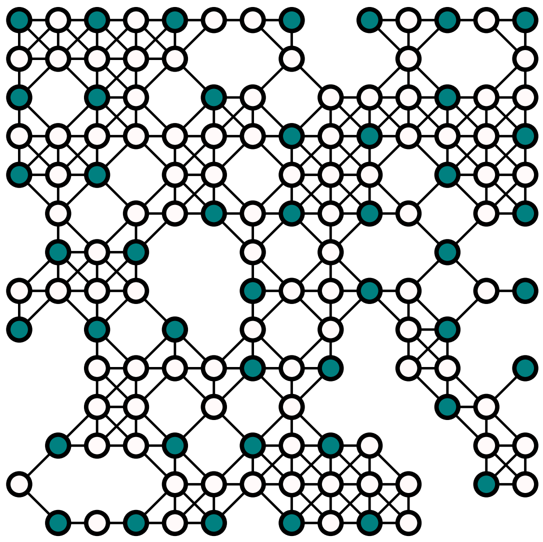


Figure 4.4: *Example QIRO instance, where teal nodes belong to the MIS* [12].

Moreover, all instances of Fig. 4.5 are randomly generated with $n = 137$. Each label's instance denotes its *hardness*, with higher labels implying greater hardness, defined as

$$\mathcal{HP} := \frac{N_{|\mathrm{MIS}|-1}}{|\mathrm{MIS}| \cdot N_{|\mathrm{MIS}|}} \tag{4.9}$$

where N_M denotes the number of independent sets of size M [12]. Such parameter has be shown to be related to the performance of both classical and quantum algorithms [133].

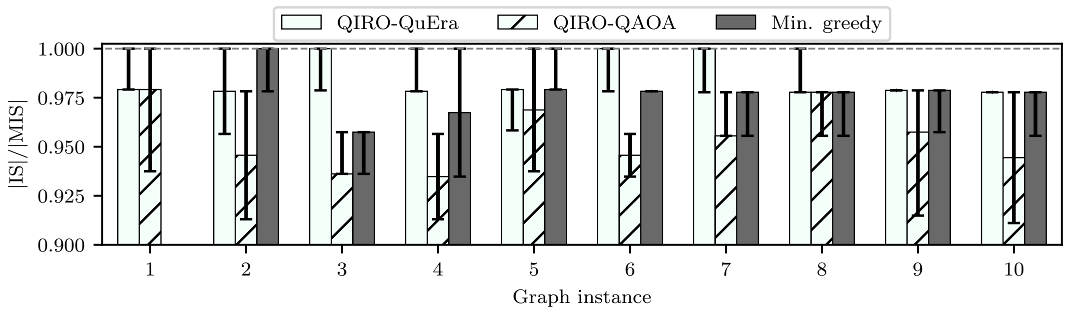


Figure 4.5: *Approximation ratio comparing the size of the independent set $|IS|$ found by QIRO with correlations from Aquila, and correlations from numerical simulations of low-depth QAOA.* [12]

As shown in the plot, quantum correlations from the QuEra machine guide the QIRO procedure more effectively than the QAOA simulations. The higher approximation ratio implies that the cardinality of the independent set found is closer to that of the true MIS. The error bars represent the best and worst solutions obtained over multiple runs for each graph instance; hence, shorter bars also indicate greater stability in the performance. Both are compared to a *minimal degree greedy algorithm*, which iteratively assigns a node with the minimal degree to the independent set and likewise removes adjacent nodes to ensure the validity of the final solution [134].

4.3 Results: Embedding Algorithm

We conclude our work by analysing the results of both the QIRO simulations and the embedding algorithm, starting with the latter. To test the embedding algorithm, we performed 1000 simulations with different seeds for each pair (N, r), with the disk radius r ranging from 0.6 to 1.4 and the system size N ranging from 8 to 24 nodes.

The choice of these parameter ranges can be better understood from Fig. 4.6, which shows a randomly generated instance with $N = 24$ nodes for three coverage radii ($r = 0.6$ on the left, $r = 1.0$ in the middle, and $r = 1.4$ on the right). As r increases, the graph connectivity quickly evolves from an almost edgeless structure to a highly connected one.

This choice is also motivated by geometric considerations. Our hexagonal shape in Python is defined with fixed coordinates, so its overall size remains constant. To prevent excessive filling of the region and the formation of clusters, we limit the system size to $N \leq 24$. Similarly, the disk radius is kept below 1.4 to avoid overly large coverage for each node.

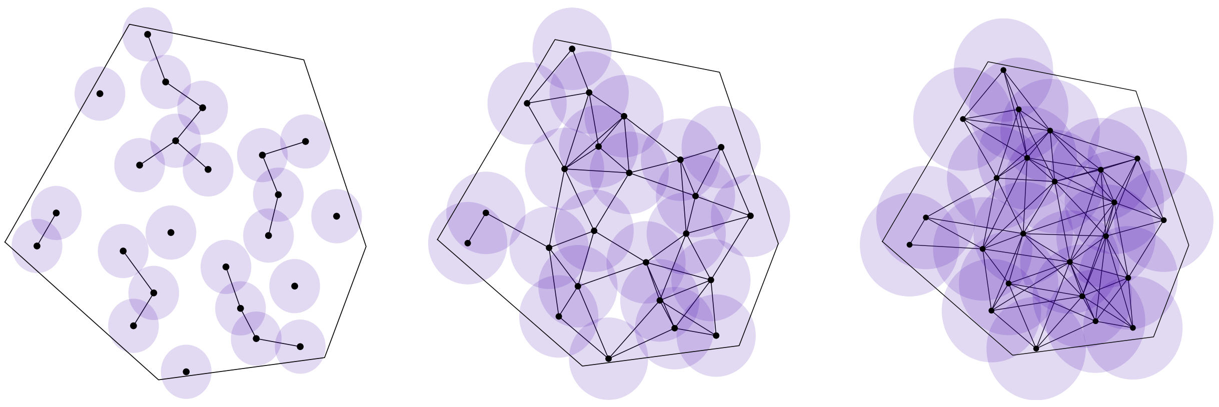


Figure 4.6: *Example of a single UDG instance (fixed seed) for three different coverage radii r:* 0.6 *on the left,* 1.0 *in the middle and* 1.4 *on the right.*

In our simulations, small radii are particularly useful for larger system sizes, where they prevent the graph from becoming excessively dense, whereas larger radii are mainly exploited at low N to ensure that the graph is connected and exhibits at least some non-trivial structure. The drawback of using small radii is that, if too few points are generated, even a uniform distribution may only partially cover the region. Most of the interesting behaviour, therefore occurs in the intermediate regime, where neither the graph is almost empty nor it is close to being complete.

With this intuitive picture in mind, we now turn to the numerical results. For clarity, Table 4.1 reports the data for the case $N = 12$ only, while the corresponding results for all other system sizes are collected in Appendix C.

The relevant parameters in the table are the following:

- r is the disk radius.
- p the success probability, i.e. the ratio between the successfully embedded instances and the total number of runs. Here, "successfully embedded" means that the algorithm was able to find a 2D configuration that preserves the original connectivity and satisfies all constraints. Conversely, "unsuccessful" means that the algorithm failed at some point during the embedding process.

- $\bar{d}^S_{2D}$ is the average *degree* of the 2D starting instances (like the ones showed in Fig. 4.6) which have been succesfully embedded.
- $\bar{d}^F_{2D}$ is the average degree of the 2D instances for which the embedding failed.
- $\bar{N}'$ is the average number of total nodes in the new 2D embedded configuration.
- $\bar{N}_W$ is the average number of *wire* nodes added during the embedding process.
- $\bar{N}_A$ is the average number of *apex* nodes (and thus of pyramids) found and embedded in each instance.
- $\bar{t}$ is the average time for the algorithm to complete the embedding.

r	p	$\bar{d}^S_{2D}$	$\bar{d}^F_{2D}$	$\bar{N}'$	$\bar{N}_W$	$\bar{N}_A$	$\bar{t}$ (s)
0.6	0.997	0.34	0.89	12.084	0.060	0.024	0.0021
0.7	0.996	0.48	0.96	12.301	0.225	0.076	0.0016
0.8	0.983	0.85	1.34	12.805	0.588	0.218	0.0016
0.9	0.961	1.16	1.66	13.988	1.473	0.514	0.0019
1.0	0.886	1.48	2.07	15.872	2.914	0.958	0.0028
1.1	0.688	1.80	2.34	18.584	5.014	1.570	0.0038
1.2	0.460	2.09	2.69	22.872	8.500	2.372	0.0076
1.3	0.214	2.32	3.03	27.509	12.229	3.280	0.0100
1.4	0.068	2.55	3.43	32.897	16.706	4.191	0.0148

Table 4.1: *Results of the embedding algorithm for $N = 12$.*

The global performance of the embedding algorithm is summarised in the success probability heat map of Fig. 4.7, where the colour encodes the fraction of successful runs as a function of the system size N and of the disk radius r.

The plot clearly shows a high-success region for small instances and relatively small radii: for $N \lesssim 14$ and $r \lesssim 0.9$ the algorithm succeeds with high probability, while the success probability rapidly decreases as either N or r increase.

Moreover, by looking at the data from the simulations, we found that there are two major problems stopping the algorithm, causing a failure of the considered instance. The first is the presence of highly clusterized structures, too complex for the algorithm to handle. The second is the failure of the path-finding function for edge-restoring, mostly due to nodes with high degree in the original instance.

This behaviour is consistent with the intuition that larger graphs and larger radii lead to much denser UDGs, and hence to more complex geometric structures for the algorithm to deal with. In particular, the almost diagonal transition from high to low success probability indicates that there is a trade-off between system size and interaction radius: for a given r there is a maximal N beyond which the embedding typically fails, and this threshold shifts to smaller N as r grows.

The same pattern emerges from the simulation data in Table 4.1 and Appendix C. In particular, the average degree of the instances for which the algorithm fails is systematically higher than that of the successfully embedded instances, most likely because highly clustered components give rise to more constrained and complex geometries.

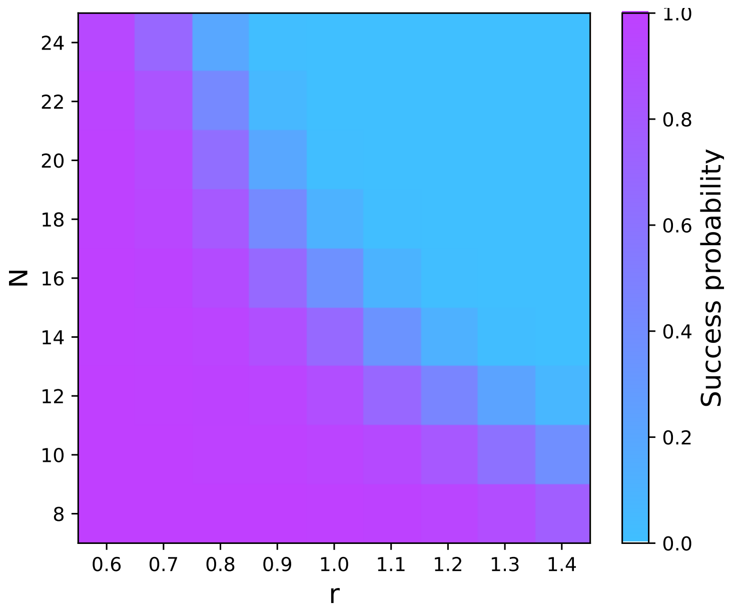


Figure 4.7: *Success probability of the embedding algorithm for each (N, r) pair. Colors show the fraction of successful runs out of 1000 seeds.*

Moreover, we notice how the algorithm starts to lose efficiency (in terms of instances' solvability) when $2 \lesssim \bar{d} \lesssim 2.5$. This behaviour is clearly illustrated in Fig. 4.8, where we plot the success probability as a function of the average 2D degree $\bar{d}_{2D}$ for all radii. All data points show a collective behaviour: for $\bar{d}_{2D} \lesssim 2$ the algorithm finds a valid embedding with high probability, while the success probability drops rapidly in the window $2 \lesssim \bar{d}_{2D} \lesssim 2.5$ and becomes essentially zero for $\bar{d}_{2D} \gtrsim 3$.

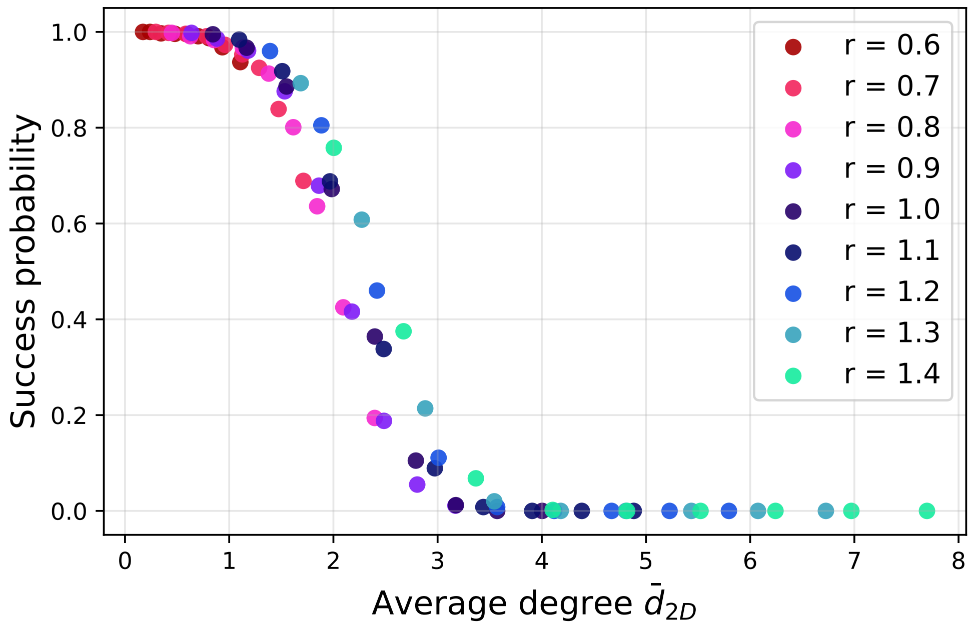


Figure 4.8: *Success probability of the embedding algorithm as a function of the average 2D degree $\bar{d}_{2D}$. Colours denote different disk radii r.*

The fact that points corresponding to different radii almost overlap indicates that the relevant control parameter for the solvability of the instances is the average degree, rather than the specific value of r or N. Dense graphs (large $\bar{d}_{2D}$) thus appear to generate geometric constraints that the current embedding algorithm can no longer handle. Moreover, in the following, our selection of instances for QIRO will be guided by the average degree of the graphs, rather than solely by N or r.

The last quantity we examine is the *scaling* of the embedding overhead, quantified by the total number of nodes in the final configuration. Figure 4.9 shows the average final size $\bar{N}'$ as a function of the original system size N for all coverage radii $r \in [0.6, 1.4]$. In Appendix C we report also two additional, more detailed plots, separating the cases $r \in [0.6, 0.9]$ and $r \in [1.0, 1.4]$.

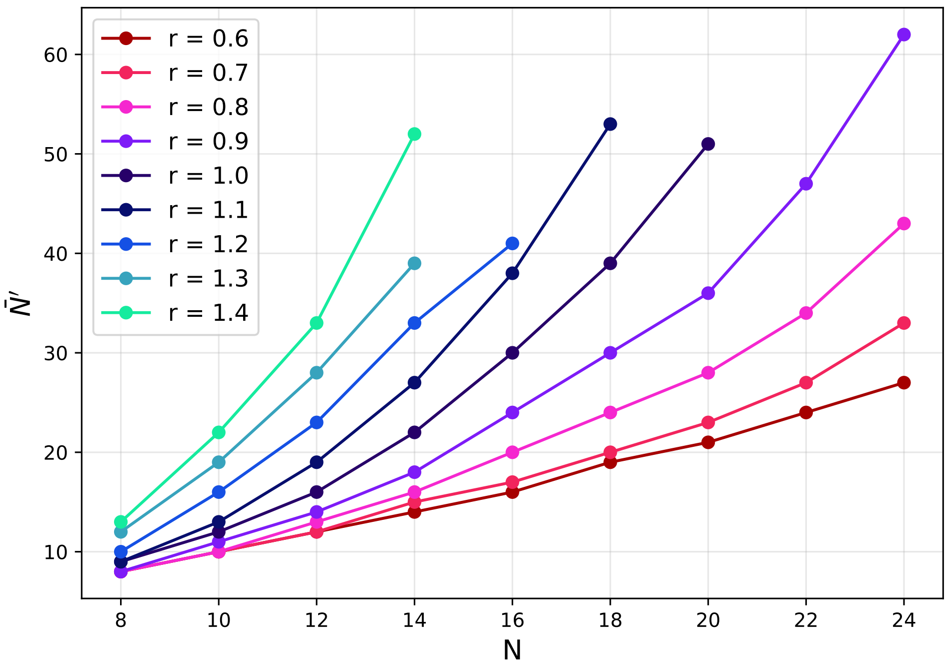


Figure 4.9: *Average system size $\bar{N}'$ of the final configuration as a function of the original system size N for all coverage radii $r \in [0.6, 1.4]$. The line connecting the markers is not a fit, just a line joining the data points, shown only to guide the eye.*

By inspecting the continuous curves connecting the data points (which are not fits, but simple lines drawn to guide the eye), we observe a much slower growth for small radii and a much faster increase for larger radii. A more detailed quantitative analysis is not possible with the limited number of available data points, and a reliable global fit would in any case require a more complex functional form. Nonetheless, the behaviour can be *approximately* regarded as linear for $r = 0.6$ and 0.7, gradually becoming polynomial for $r \in [0.7, 0.9]$, and eventually approaching an exponential-like growth for $r > 1.0$. Simple χ^2 tests are consistent with this qualitative picture.

4.4 Results: Simulations with QIRO

We are now ready to assess the efficiency of QIRO on the instances produced by the embedding algorithm. In all subsequent analyses, for each system size N we select the 20 instances with the largest final size N' and average over this subset. This yields a total of 180 embedded instances for each plot. This choice is motivated by the fact that the largest embedded instances (in terms of final size N') also correspond to the highest average degrees that the algorithm can handle, placing us in the most challenging and thus most informative regime to analyse.

In addition, the choice of these particular instances is motivated by the working assumption, discussed in Sec. 3.3, that adding quantum wires does not alter the system's ground state. Focusing on the largest instances provides a stringent test: if the original solution remains correctly encoded even after introducing a very large number of auxiliary nodes, then smaller instances, which require fewer wires, are expected to behave at least as reliably.

Furthermore, focusing on these larger instances captures precisely the regime in which the classical brute-force search we use could require prohibitively long runtimes, and where quantum computers could provide a genuine advantage. In this paragraph, we analyse and compare the performance of both approaches.

Before delving into the details, we emphasize that the Hamiltonian used by QIRO is *uniform*: the algorithm effectively solves an *unweighted* version of the MIS, with all node weights, i.e. the h_i in Eq. (4.7), set to 1 by default. Moreover all the J_{ij} are set equal to the penalty factor λ, provided as input of the algorithm, thus giving the same penalty to all indepnendence constraints. In other words, the cost function only distinguishes configurations by the number of selected vertices, and not by heterogeneous weights.

To have a classical benchmark as comparison, we compute the MIS on the *original* 2D instance via a brute-force function, which exploits the well-known duality between independent sets and *cliques* (subset of vertices of an undirected graph where every two distinct vertices are adjacent): an independent set in a graph G is exactly a clique in its *complement* $\overline{G}$, where two vertices are adjacent if and only if they are non-adjacent in G.

To enumerate all maximal independent sets of G, we therefore build $\overline{G}$ and apply the *Bron-Kerbosch algorithm* [135], a classic recursive backtracking procedure that efficiently lists all maximal cliques of a graph by growing candidate cliques and pruning via neighborhood intersections (with pivoting heuristics in its optimized form). Each maximal clique returned by Bron-Kerbosch on $\overline{G}$ corresponds to a maximal independent set in G; selecting those of largest cardinality then yields all MISs of the original graph.

To assess QIRO's efficiency we consider three parameters: the **convergence time**, the **approximation ratio** [136] and the **binary fidelity**, which we now introduce starting from the first. The *convergence time*, which is simply defined as the wall-clock runtime (in seconds) required to obtain a solution. We measure it both for QIRO, run on the final embedded instances, and for the classical brute-force search on the original UDGs.

Fig. 4.10 shows the average convergence time of the brute-force search (left) and the average convergence time of QIRO on the same instances, evaluated on the final embedded graphs after adding slack and wire nodes (right). Error bars indicate one standard deviation over the sample of 20 largest embedded instances for each N.

As expected, the brute-force search is extremely fast for small instances and quickly exhibits the characteristic exponential growth of the runtime. By contrast, QIRO maintains a high efficiency even for the largest instances we consider. We note that the x-axis shows only the number of physical nodes as a common reference for the same set of instances before and after embedding, while the embedded graphs can contain up to ~ 100 atoms.

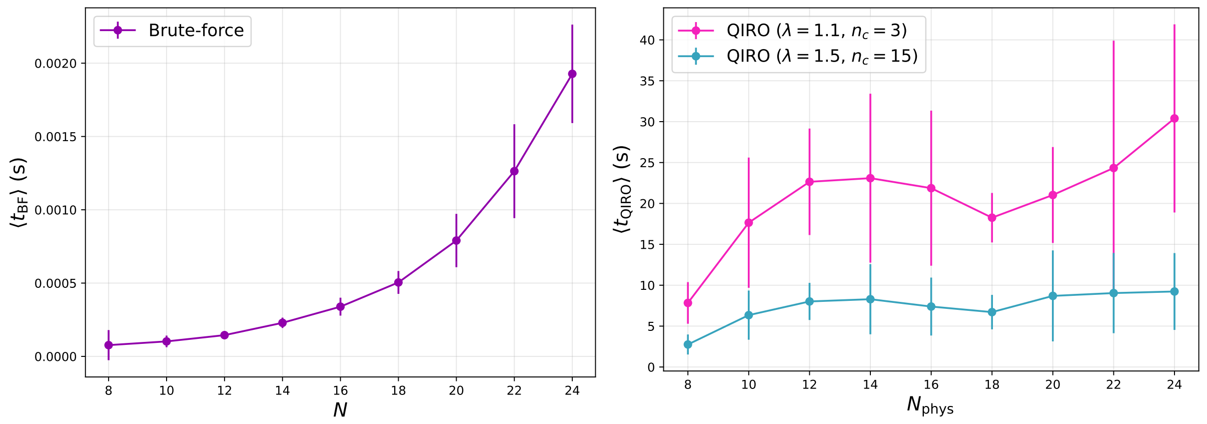


Figure 4.10: *Average convergence time as a function of the number of physical nodes N for the classical brute-force MIS search on the original graphs (left) and for QIRO on the corresponding embedded instances (right). n_c denotes the maximum size of any connected component in the graph that is to be solved by brute force.*

In our simulations we used two parameter sets for QIRO: $\lambda = 1.1$ and $n_c = 3$, matching the example Jupyter notebook provided with the original GitHub code, and $\lambda = 1.5$ and $n_c = 15$, as in [12]. The latter choice, which slightly increases the penalty parameter λ of Eq. (4.6) and allows larger connected components to be solved classically, leads to a substantial speed-up and to more uniform performance across instances, as reflected by the smaller standard deviations.

Although the brute-force search is very fast on the small graphs we benchmark, it only works at that scale. For the much larger embedded instances that QIRO solves in a few seconds, the same exponential brute-force search would require astronomically long runtimes, effectively making it unusable.

Indeed, the worst-case running time of the Bron-Kerbosch algorithm scales as $\mathcal{O}(3^{n/3})$, in terms of the number of recursive calls. We can therefore use this bound to estimate the runtime at larger N as follows. For $N = 24$, the upper bound is $3^{24/3} = 3^8 \approx 6.5 \times 10^3$ recursive calls. Since in this case the function finds the MIS in about 0.0020 s, the algorithm performs roughly $6.5 \times 10^3/0.0020 \approx 3.25 \times 10^6$ calls per second.

For $N = 50$, the bound becomes $3^{50/3} \approx 9 \times 10^7$ calls, corresponding to a runtime of about 28 s, which is still acceptable. However, for $N = 60$ the estimated runtime already grows to ~ 18 minutes, and for $N = 100$ it reaches an astonishing ~ 78 years.

A final note on the running times is that, on the plots shown in Fig. 4.10, the classical brute-force search is clearly faster, with values that are up to three orders of magnitude smaller than those of QIRO. However, the key result is that, unlike brute force, QIRO does not exhibit exponential scaling: in our simulations its efficiency remains remarkably regular and uniform across the tested system sizes.

Moving to the second parameter, we define the approximation ratio as

$$a = \frac{E_{\text{QIRO}}}{E_{\text{Exact}}} \tag{4.10}$$

where E_{Exact} is the energy of the bitstring corresponding to the exact solution, obtained by brute-force search, and E_{QIRO} is the energy of the bitstring returned by QIRO. In both cases the energy is computed as

$$E = -\sum_i S_i + \sum_{\langle ij \rangle} S_i S_j \tag{4.11}$$

with $S_i \in \{0, 1\}$ and where $\langle ij \rangle$ denotes pairs of nearest-neighbour nodes, which in our case are nodes connected by an edge of the underlying UDG. We choose to define a in this way so that $a = 1$ corresponds to optimal performance and $a < 1$ quantifies the relative energy overhead of QIRO with respect to the exact optimum.

Fig. 4.11 shows the average energies E_{Exact} and E_{QIRO} as a function of N (left), and the corresponding approximation ratio (4.10) (right). Error bars indicate again one standard deviation over the 20 embedded instances used for each system size.

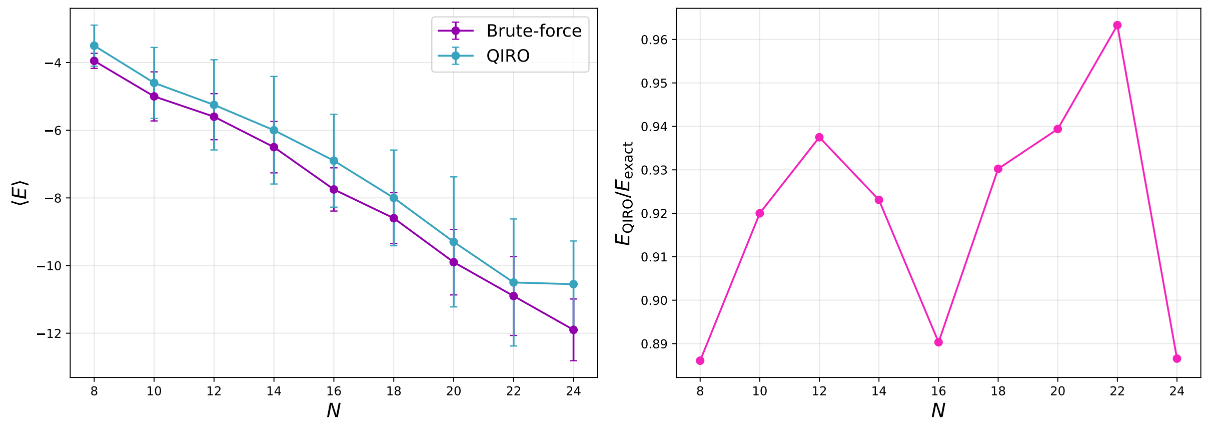


Figure 4.11: *Average energies obtained with brute-force and QIRO (left) and corresponding approximation ratio E_{QIRO}/E_{Exact} (right) as a function of the system size N.*

In the right panel of Fig. 4.11, the error bars are not explicitly shown. The uncertainties were nonetheless computed by standard error propagation, treating the errors on the numerator and denominator as independent. In all cases, the relative errors on the ratio lie in the 15-25 % range. We also observe a peculiar drop in the approximation ratio at system sizes that are multiples of $N = 8$. The precise origin of this effect is unknown and would require a more detailed analysis, but it may simply be due to statistical fluctuations.

The energies obtained with QIRO are systematically slightly higher than the exact ground-state energies found by the brute-force solver, but the discrepancy remains small in all instances. This confirms that the introduction of auxiliary nodes in our construction does not alter the optimal solution or the structure of the ground state in any substantial way: the enlarged graph still encodes the original problem with high fidelity.

In particular, the approximation ratio between the QIRO solution and the brute-force optimum (as defined above) takes values in the range from ~ 0.89 to ~ 0.96, i.e., very close to unity. Such high ratios indicate that QIRO recovers most of the optimal objective value, so that the combined effect of the auxiliary-node mapping and the low-depth variational ansatz only leads to a moderate loss in performance. Overall, within the regime of system sizes and connectivities considered here, QIRO thus behaves as a high-quality heuristic rather than a crude approximation.

As third parameter, we introduce the *binary fidelity* F_B, which quantifies how well the bitstring returned by QIRO matches any of the brute-force exact MIS solutions.

Let S^Q denote the bitstring produced by QIRO, and let $\{S^{B,(\alpha)}\}_{\alpha=1}^{n(S^B)}$ be the set of all optimal MIS bitstrings of common cardinality $C(S^B)$. Each bitstring $S^{B,(\alpha)}$ has length N, and each entry $S_i \in 0, 1$ indicates whether node i is included in the MIS.

Since multiple MIS configurations typically exist, comparing S^Q with any particular representative may underestimate the actual performance of QIRO. We therefore define the fidelity *with respect to the closest MIS configuration*, i.e. we consider the maximal overlap between S^Q and any optimal MIS bitstring:

$$F_B^* = \frac{1}{C(S^B)} \max_{\alpha=1,\dots,n(S^B)} \sum_{i=1}^{N} S_i^Q S_i^{B,(\alpha)} \tag{4.12}$$

where the quantity

$$\sum_{i=1}^{N} S_i^Q S_i^{B,(\alpha)} \tag{4.13}$$

counts the number of nodes set to 1 both in the QIRO solution and in the α-th exact MIS configuration, i.e. the MIS nodes correctly identified by QIRO in that optimum. Dividing by the MIS cardinality $C(S^B)$ ensures that $F_B \in [0, 1]$.

Thus, F_B measures the fraction of MIS nodes correctly found by QIRO, maximised over all degenerate MIS configurations. By construction, $F_B = 1$ if and only if QIRO returns at least one exact MIS configuration, so degeneracy does not artificially reduce the fidelity.

The binary fidelity F_B is the analogue of the *spin-overlap order parameter* [137]

$$q = \frac{1}{N} \sum_i S_i^{(A)} S_i^{(B)} \tag{4.14}$$

used to quantify the similarity between two spin configurations; the only difference is that here we explicitly account for the degeneracy of the ground state by maximising the overlap over all optimal MIS configurations.

Fig. 4.12 shows the behaviour of the average binary fidelity F_B as a function of N (as usual, error bars denote one standard deviation). Across all system sizes, F_B remains in the range from ~ 0.7 to ~ 0.85, denoting that QIRO typically identifies a large fraction of the MIS nodes in at least one optimal configuration.

This confirms that, beyond achieving near-optimal energies, the algorithm also reconstructs the combinatorial structure of the solution with reasonably high accuracy: only about 15-30 % of MIS nodes are, on average, misplaced or missed. Since F_B is maximised over all degenerate MIS configurations, these values provide a stringent node-by-node measure of performance that is consistent with the high approximation ratios reported in Fig. 4.11.

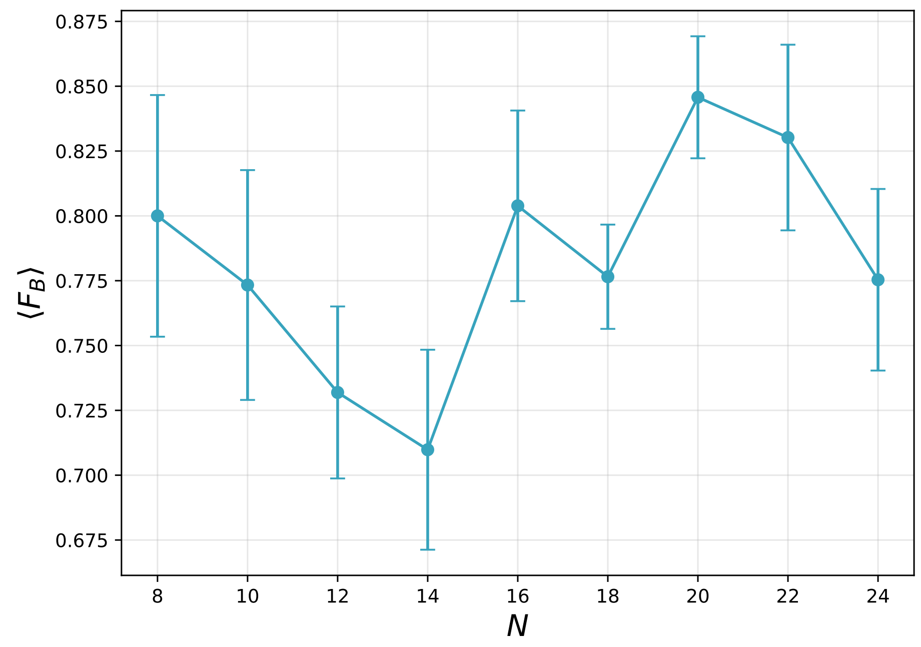


Figure 4.12: *Average Binary fidelity F_B between the QIRO solution and an exact MIS configuration as a function of the system size N.*

Conclusions

In this work we have proposed a method for solving three-dimensional constrained QUBO instances arising from the MIS problem on unit disk graphs (UDGs). At the core of our approach lies a geometric *embedding algorithm* that maps three-dimensional instances into two-dimensional, topologically equivalent UDGs with only nearest-neighbour interactions, making them compatible with neutral-atom quantum devices based on Rydberg-atom arrays.

The embedding algorithm is built upon the core idea that a sufficiently regular three-dimensional pyramid with triangular base can be mapped to a two-dimensional, bow-tie-like structure. It is built on a systematic analysis of the possible multi-pyramid configurations and of how to preserve the original connectivity while introducing as few auxiliary nodes as possible.

We analysed the performance of this embedding procedure in terms of its success probability and of the scaling of the final graph size with the original system size and connectivity. We found that the algorithm has a high success probability when the starting instance has an average degree $\bar{d}_{2D} \lesssim 2$, whereas in the range $2 \lesssim \bar{d}_{2D} \lesssim 3$ the success probability rapidly drops to zero. This indicates that the algorithm can be scaled to larger system sizes N, provided that the average degree remains within this threshold.

We then analysed how the size of the embedded graph scales with the original system size as a function of the disk radius r. A qualitative comparison of different trial fits indicates that, for relatively small radii, the scaling is approximately linear or at most low-order polynomial. For larger radii, the stronger clustering of the initial instances instead leads to quadratic or nearly exponential growth.

The MIS problem on the embedded graphs was tackled using QIRO, a variational hybrid quantum-classical algorithm of the QAOA family. By comparing its performance to a classical brute-force solver, we showed that QIRO can handle relatively large system sizes ($N \sim 100$) with high efficiency in terms of both convergence time and approximation ratio. The near-optimal energies and high approximation ratios confirm that the embedding process does not alter the ground state of the original Hamiltonian and that the enlarged graph still encodes the correct solution with high fidelity.

Finally, the analysis of the binary fidelity shows that QIRO typically returns a solution bitstring that reconstructs the exact MIS configuration with good accuracy, missing or misplacing at most about 30 % of the MIS nodes. Overall, our results indicate that the combination of geometric embedding and QIRO provides a viable route towards solving three-dimensional constrained QUBO instances on Rydberg-atom quantum devices.

Looking ahead, several directions emerge quite naturally from this work. On the classical side, the embedding algorithm could first be stress-tested on larger three-dimensional instances at fixed average degree, in order to better characterise its scalability in realistic regimes. Beyond this, one may extend its capabilities by enriching the library of geometric gadgets used to embed more complex local structures, thereby pushing the maximum average degree for which the algorithm remains successful. In parallel, the path-finding routine used to reconstruct edges could be optimised to reduce the number of wire nodes, for example by systematically favouring shorter or less congested routes, thus limiting the overhead in the final embedded graph.

On the quantum side, QIRO could be refined in at least two complementary ways. A natural extension is to implement the *weighted* version of MIS and test whether the correct solution is still encoded with high probability in the presence of non-uniform costs. In addition, exploring deeper QAOA-style circuits, while keeping classical simulation or hardware execution efficient, may further improve approximation ratios and binary fidelities, thereby enhancing the overall performance of the combined embedding-QIRO pipeline.

A

Quantum Mechanics

A.1 Postulates of Quantum Mechanics

Quantum algorithms are formulated within the rigorous mathematical framework of *quantum mechanics* (QM). First, it's necessary to establish the fundamental mathematical and physical concepts underlying the theory. This chapter provides a concise overview of the formalism of quantum mechanics, expressed in the language of **Dirac notation** (also called **bra-ket notation**), which we briefly recall here. The exposition of this chapter follows the treatment of [44]. In Dirac notation, every column vector of the Hilbert space $\mathcal{H}$[1] that represents a physical quantum state is denoted by a **ket** vector:

$$|v\rangle = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix} \in \mathcal{H} \tag{A.1}$$

To each ket vector, one can associate an element of the *dual space* $\mathcal{H}^*$, called a **bra**, which corresponds to the *conjugate transpose* (*Hermitian adjoint*) of the ket:

$$\langle v| = \big(|v\rangle \big)^\dagger = \begin{pmatrix} v_1^* & v_2^* & \cdots & v_n^* \end{pmatrix} \in \mathcal{H}^* \tag{A.2}$$

The *scalar product* between two vectors $|v\rangle$ and $|w\rangle$ is then expressed as

$$(w, v) = \langle w|v\rangle = \begin{pmatrix} w_1^* & w_2^* & \cdots & w_n^* \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix} = \sum_n w_n^* v_n \in \mathbb{C} \tag{A.3}$$

With these foundations in place, we can now state the *first postulate* of QM:

> **Postulate 1:** Associated to any isolated physical system is a complex vector space with inner product (that is, a Hilbert space $\mathcal{H}$) known as the state space of the system. The system is completely described by its state vector $|\psi\rangle$, which is a unit vector in $\mathcal{H}$, namely $\langle\psi|\psi\rangle = 1$.

The simplest quantum system we can describe is the **qubit**, a two-dimensional quantum system whose basis vectors are conventionally denoted by $|0\rangle$ and $|1\rangle$. These states satisfy

$$\begin{cases} \mathcal{H} = \text{span}\Big\{ |0\rangle \,, |1\rangle \Big\} \\ \langle 0|0\rangle = \langle 1|1\rangle = 1 \\ \langle 0|1\rangle = \langle 1|0\rangle = 0 \end{cases} \tag{A.4}$$

[1] For simplicity, we will focus on *finite-dimensional* Hilbert spaces, i.e. those with $\dim(\mathcal{H}) < \infty$.

In the most general case a **qubit state** will be a *linear superposition* of the form

$$\begin{cases} |\psi\rangle = a\,|0\rangle + b\,|1\rangle \\ |a|^2 + |b|^2 = 1 \\ a, b \in \mathbb{C} \end{cases} \tag{A.5}$$

where a and b are called the *amplitudes* of the states $|0\rangle$ and $|1\rangle$ respectively.

The next natural question concerns the time evolution of a quantum state: how does the state $|\psi\rangle$ of a system change with time? This is precisely addressed by the second postulate, which prescribes the law governing state evolution.

> **Postulate 2:** The evolution of a closed[2] quantum system is described by the **Schrödinger equation**
>
> $$i\hbar\frac{d}{dt}|\psi(t)\rangle = H|\psi(t)\rangle \tag{A.6}$$
>
> where H is the (Hermitian) Hamiltonian operator. The evolution of the system is **unitary**, that is
>
> $$\begin{aligned} &|\psi(t)\rangle = \mathcal{U}(t)\,|\psi(0)\rangle \\ &\mathcal{U}(t) = \exp\big(-iHt/\hbar\big) \\ &\mathcal{U}^\dagger(t)\,\mathcal{U}(t) = \mathbb{I} \end{aligned} \tag{A.7}$$

This unitary evolution is the theoretical basis for the notion of *quantum gates*, which implement such transformations in the circuit model of quantum computation as we shall see. Having established how quantum states evolve in time, we now turn to the question of how information is extracted from it, which is addressed by the third postulate.

> **Postulate 3:** Quantum measurements are described by a collection $\{M_m\}$ of measurement operators, which act on the state space of the system being measured. The index m refers to the measurement outcomes that may occur in the experiment. If the quantum system's state is $|\psi\rangle$ immediately before the measurement then the probability that result m occurs is given by
>
> $$p(m) = \langle\psi|\, M_m^\dagger M_m\,|\psi\rangle \tag{A.8}$$
>
> and the state of the system after the measurement is
>
> $$|\psi'\rangle = \frac{M_m\,|\psi\rangle}{\sqrt{p(m)}} \tag{A.9}$$
>
> with the measurement operators satisfying the completeness equation
>
> $$\sum_m M_m^\dagger M_m = \mathbb{I} \tag{A.10}$$

[2] A system that doesn't interact with any other system. Strictly speaking, no physical system is perfectly closed, but many can be approximated as such, with their dynamics well described by unitary evolution.

An important special case of quantum measurements is given by **projective measurements**, which are directly associated with **observables**. These are represented by Hermitian (self-adjoint) operators $A = A^\dagger$ acting on the Hilbert space $\mathcal{H}$.

The Hermiticity of observables is crucial: by the *spectral theorem*, such operators possess real eigenvalues a_i and a complete orthonormal set of eigenvectors $\{|a_i\rangle\}$ that form a basis of $\mathcal{H}$. Consequently, the operator admits the *spectral decomposition*

$$A = \sum_i a_i \, |a_i\rangle\!\langle a_i| \tag{A.11}$$

Performing a *projective measurement* of an observable A is described by the set of **projectors** $\{P_i = |a_i\rangle \langle a_i|\}$, which constitute a special case of the general measurement operators M_m introduced above in Postulate 3. If the system is in the state $|\psi\rangle$, the probability of obtaining the outcome a_i is then given by

$$p(a_i) = \langle\psi| P_i |\psi\rangle = |\langle a_i|\psi\rangle|^2 \tag{A.12}$$

while the post-measurement state reads

$$|\psi'\rangle = \frac{P_i |\psi\rangle}{\sqrt{p(a_i)}} = |a_i\rangle \tag{A.13}$$

This construction highlights the role of the *outer product* $|a_i\rangle \langle a_i|$ as the projector onto the eigenspace corresponding to the eigenvalue a_i. More generally, outer products of the form $|\phi\rangle \langle\psi|$ define linear operators that map states to states, and they play a fundamental role in the operator formalism of quantum mechanics.

Having examined the formulation of measurements in quantum mechanics, we now turn to the fourth postulate, which specifies how composite systems are represented.

> **Postulate 4:** The state space of a composite physical system $\mathcal{H}_{\text{tot}}$ is the tensor product of the state spaces of the component physical systems, namely $\mathcal{H}_{\text{tot}} = \mathcal{H}_1 \otimes \mathcal{H}_2 \otimes \cdots \otimes \mathcal{H}_n$. Moreover, if we have systems numbered 1 through n, and system number i is prepared in the state $|\psi_i\rangle$, then the joint state of the total system is $|\psi_{tot}\rangle = |\psi_1\rangle \otimes |\psi_2\rangle \otimes \cdots \otimes |\psi_n\rangle \equiv |\psi_1\psi_2 \ldots \psi_n\rangle$.

This postulate not only defines how to construct the state space of *multi-partite systems*, but also introduces the possibility of **entanglement**.

> **Def. A.1.** Let $|\psi_{AB}\rangle \in \mathcal{H}_A \otimes \mathcal{H}_B$. We say that $|\psi_{AB}\rangle$ is a **product state** if $\exists$ $|\phi_A\rangle \in \mathcal{H}_A$ and $|\chi_B\rangle \in \mathcal{H}_B$ such that
>
> $$|\psi_{AB}\rangle = |\phi_A\rangle \otimes |\psi_B\rangle \tag{A.14}$$
>
> We say that $|\psi_{AB}\rangle$ is **entangled** otherwise

Entanglement is a genuinely non-classical feature of quantum mechanics and constitutes a key resource for quantum information processing, underpinning tasks such as quantum teleportation, Bell inequality violations, and the advantage of many quantum algorithms.

Up to this point, we have presented the standard formulation of the postulates of QM. In the next section, we introduce a more general framework based on **density operators**.

A.2 Density Operators

The formulation in terms of **density operators** is mathematically equivalent to the standard approach, yet often provides a more versatile tool. This framework becomes particularly valuable when dealing with systems whose state is not completely specified.

More precisely, suppose a quantum system can be found in one of several **pure states** $|\psi_i\rangle$, each occurring with probability p_i. The collection $\{p_i, |\psi_i\rangle\}$ is called an *ensemble of pure states.* The corresponding density operator of the system is defined as

$$\rho \equiv \sum_i p_i \, |\psi_i\rangle\langle\psi_i| \tag{A.15}$$

The density operator is also commonly referred to as the **density matrix**; we will therefore use the two terms interchangeably. In particular, for ρ to qualify as a density operator associated with some ensemble $\{p_i, |\psi_i\rangle\}$, it must satisfy two properties:

- ρ must have **unit trace**, namely $\mathrm{Tr}[\rho] = 1$.
- ρ must be **positive semidefinite**, i.e. $\rho \geq 0$, meaning that all its eigenvalues are non-negative, or equivalently that $\langle\psi| \, \rho \, |\psi\rangle \geq 0 \; \forall \, |\psi\rangle$.

Before reformulating QM's postulates in this formalism, it is helpful to introduce some terminology and a useful property of density operators. First, if a quantum system is in a *pure state*[3] $|\psi\rangle$, the corresponding density operator takes the simple form

$$\rho = |\psi\rangle\langle\psi| \tag{A.16}$$

More generally, the system may be in a **mixed state**[4], representing a statistical ensemble of possible pure states ρ_i, each occurring with probability p_i. In this case,

$$\rho = \sum_i p_i \, |\psi_i\rangle\langle\psi_i| = \sum_i p_i \rho_i \tag{A.17}$$

A pure state is the special case where all $p_i = 0$ except one, which equals 1, meaning that the system's quantum state $|\psi\rangle$ is *completely determined.* In contrast, a mixed state reflects probabilistic uncertainty over a set of possible pure states.

The mathematical distinction between pure and mixed states can be captured by a simple criterion, summarized in the following proposition.

Prop. A.1. Given a state $\rho \in \mathcal{S}(\mathcal{H})$ (the set of all quantum states in $\mathcal{H}$) it is $\mathrm{Tr}[\rho^2] \leq 1$. Moreover $\mathrm{Tr}[\rho^2] = 1$ if and only if ρ is pure.

The quantity $\mathcal{P}(\rho) = \mathrm{Tr}[\rho^2]$ is therefore called the **purity** of the state ρ.

[3] In the literature, the term "pure state" is often used to refer only to the state vector $|\psi\rangle$ itself, in order to distinguish it from the corresponding density operator ρ.

[4] Sometimes ρ is referred to as the (statistical) *mixture* of the states ρ_i with probabilities p_i.

At this point, we can reformulate QM's postulates within this framework.

Postulate 1: Associated to any physical system is a complex vector space with inner product (that is, a Hilbert space $\mathcal{H}$), known as the state space of the system. The system is completely described by its density operator, which is a positive operator ρ with trace one, acting on the state space of the system. If a quantum system is in the state ρ_i with probability p_i, then the density operator for the system is

$$\rho = \sum_i p_i \rho_i \tag{A.18}$$

Having established the description of states, we can turn to their evolution.

Postulate 2: The evolution of a closed quantumm system is described by a unitary transformation. That is, the state ρ of the system at time t_1 is related to the state ρ' of the system aat time t_2 by a unitary operator $\mathcal{U}$, which depends only on the times t_1 and t_2. More formally

$$\rho \xrightarrow{\mathcal{U}} \mathcal{U}\rho\mathcal{U}^\dagger = \sum_i p_i \mathcal{U} |\psi_i\rangle\langle\psi_i| \mathcal{U}^\dagger = \sum_i p_i |\psi_i'\rangle\langle\psi_i'| = \rho' \tag{A.19}$$

In the same spirit, we now restate the third postulate.

Postulate 3: Quantum measurements are described by a collection $\{M_m\}$ of measurement operators. These are operators acting on the state space of the system being measured. The index m refers to the measurement outcomes that may occur in the experiment. If the state of the quantum system is ρ immediately before the measurement then the probability that result m occurs is given by

$$p(m) = \mathrm{Tr}\left[M_m^\dagger M_m \rho\right] \tag{A.20}$$

and the state of the system after the measurement is

$$\rho' = \frac{M_m \rho M_m^\dagger}{p(m)} \tag{A.21}$$

The measurement operators satisfy the completeness equation

$$\sum_m M_m^\dagger M_m = \mathbb{I} \tag{A.22}$$

Finally, we turn to the fourth postulate.

Postulate 4: The state space of a composite physical system is the tensor product of the state spaces of the component physical systems, namely $\mathcal{H}_{\text{tot}} = \mathcal{H}_1 \otimes \mathcal{H}_2 \otimes \cdots \otimes \mathcal{H}_n$. Moreover, if we have systems numbered 1 through n, and system number i is prepared in the state ρ_i, then the joint state of the total system is $\rho_1 \otimes \rho_2 \otimes \cdots \otimes \rho_n$.

As anticipated, this reformulation of the fundamental postulates in terms of density operators is mathematically equivalent to the usual description with state vectors.

We can also reformulate the definition of entangled states within this formalism.

Def. A.2. Let $\rho \in \mathcal{S}(\mathcal{H}_A \otimes \mathcal{H}_B)$. Then we say that

- ρ is a **product state** if $\exists \sigma \in \mathcal{S}(\mathcal{H}_A)$ and $\omega \in \mathcal{S}(\mathcal{H}_B)$ such that $\rho = \sigma \otimes \omega$.
- ρ is a **separable state** if $\exists \{\rho_k^A\}$, $\{\rho_k^B\}$ and $p_k \geq 0$ with

$$\sum_k p_k = 1 \tag{A.23}$$

  such that

$$\rho = \sum_k p_k \rho_k^A \otimes \rho_k^B \tag{A.24}$$

- ρ is **entangled** otherwise.

An important property of $\mathcal{S}(\mathcal{H})$ is that it is **convex**, that is, taken ρ_1, $\rho_2 \in \mathcal{S}(\mathcal{H})$

$$\rho(\lambda) = \lambda \rho_1 + (1 - \lambda)\rho_2 \in \mathcal{S}(\mathcal{H}) \tag{A.25}$$

for all $0 \leq \lambda \leq 1$. This property is illustrated schematically in Fig. A.1.

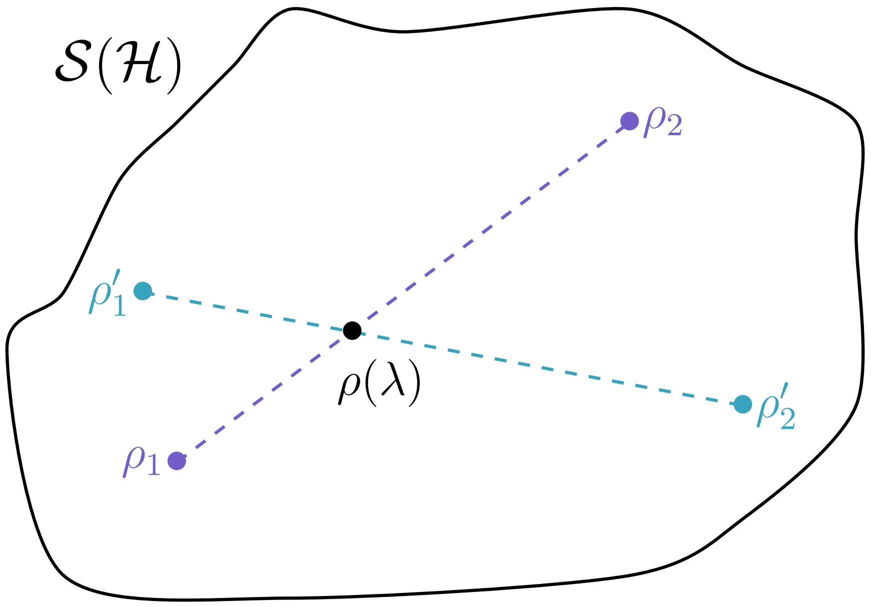


Figure A.1: *Given ρ_1, $\rho_2 \in \mathcal{S}(\mathcal{H})$, any point on the line connecting them is a valid density operator and can be written as in (A.25). The same density operator $\rho(\lambda)$ can also be expressed in different ways (non-uniqueness of representation), e.g. $\rho(\lambda) = \lambda\,\rho_1 + (1-\lambda)\rho_2 = \mu\,\rho_1' + (1-\mu)\rho_2'$. If instead $\rho(\lambda)$ is pure (points on the boundary of $\mathcal{S}(\mathcal{H})$, where no interpolation is possible), its representation is unique, namely $\rho_1 = \rho_2 = |\psi\rangle\langle\psi|$.*

A.3 The Bloch Representation

The general qubit state (A.5) can be expressed in the form (up to an overall phase[5])

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + \exp(i\varphi)\sin\left(\frac{\theta}{2}\right)|1\rangle\,. \tag{A.26}$$

With $0 \le \theta \le \pi$ and $0 \le \varphi < 2\pi$. The parameters θ and φ specify a point on the surface of the unit three-dimensional sphere, depicted in Fig. A.2. This sphere, known as the **Bloch sphere**, provides a convenient way to visualize the state of a single qubit. Moreover, many single-qubit operations are naturally described within this framework. Nonetheless, one must keep in mind that this intuition has limitations, as no straightforward generalization of the Bloch sphere exists for systems of multiple qubits.

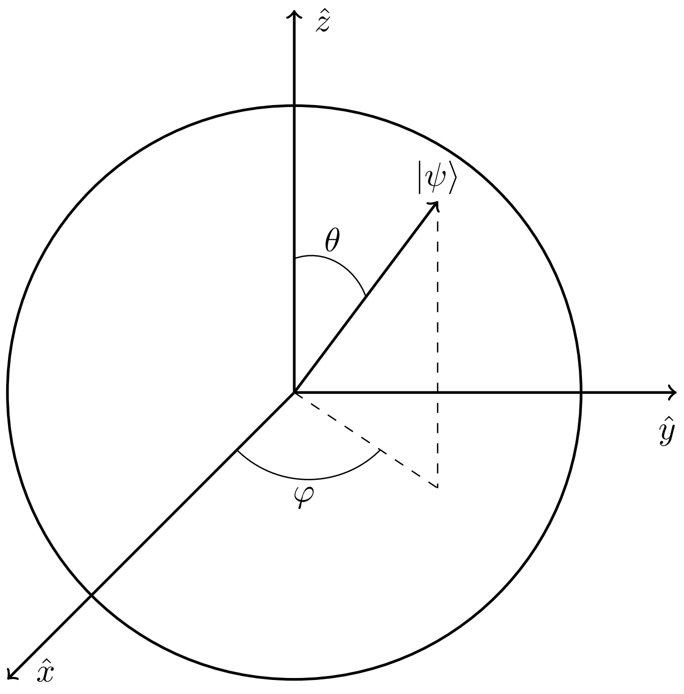


Figure A.2: *Bloch sphere representation of a qubit.*

Pure states are represented on the surface of the Bloch sphere, while mixed states are depicted as points inside it. To provide a unified notation for visualizing both pure and mixed states on the sphere, we introduce the well-known **Bloch representation**. Before delving into it, we first need to introduce the **Pauli matrices**

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad \sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \quad \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{A.27}$$

These will be grouped into the following vector:

$$\vec{\sigma} = (\sigma_x, \sigma_y, \sigma_z) \tag{A.28}$$

[5] This global phase is always omitted, as it has no observable physical effects.

Since the Pauli matrices, along with the identity matrix $\mathbb{I}$, form a basis for the space of 2×2 Hermitian matrices over the real numbers, it follows that any single-qubit state can be expressed as a linear combination of the form:

$$\rho = a_0\mathbb{I} + a_1\sigma_x + a_2\sigma_y + a_3\sigma_z \tag{A.29}$$

If we now introduce the so-called **Bloch vector**[6]

$$\vec{r} = (2a_1, 2a_2, 2a_3) \tag{A.30}$$

Then we can write the famous *Bloch representation* as

$$\rho = \frac{1}{2}\big(\mathbb{I} + \vec{r} \cdot \vec{\sigma}\big) \tag{A.31}$$

where $\|\vec{r}\| \leq 1$, with equality holding for pure states. For two distinct pure states ρ_1 and ρ_2 with corresponding Bloch vectors $\vec{r}_1$ and $\vec{r}_2$, a mixed state can be written using an expression analogous to the *convex representation* (A.25):

$$\rho(\lambda) = \lambda\rho_1 + (1-\lambda)\rho_2 = \frac{1}{2}\Big(\mathbb{I} + \vec{r}(\lambda) \cdot \vec{\sigma}\Big) \tag{A.32}$$

where $\vec{r}(\lambda)$ lies in the interior of the sphere and reads

$$\vec{r}(\lambda) = \lambda\vec{r}_1 + (1-\lambda)\vec{r}_2 \tag{A.33}$$

This situation is shown in Fig. A.3.

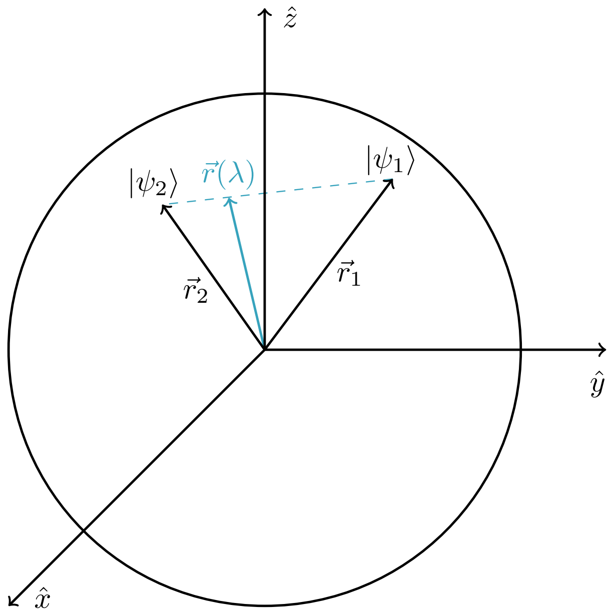


Figure A.3: *Mixed state's Bloch vector.*

[6] The value of a_0 can be proven to be $1/2$, that's why it is not included in the Bloch vector.

A.4 Reduced Density Operators

The most profound use of the density operator lies in its role as a descriptive tool for *subsystems* of a composite quantum system. This role is fulfilled by the **reduced density operator**. Consider two physical systems, A and B, jointly described by a density operator ρ^{AB}. The reduced density operator of system A is then defined by

$$\rho^A \equiv \mathrm{Tr}_B\left[\rho^{AB}\right] \tag{A.34}$$

where Tr_B is a map of operators known as the **partial trace** over system B, defined by

$$\mathrm{Tr}_B = \Big[\,|a_1\rangle\langle a_2| \otimes |b_1\rangle\langle b_2|\,\Big] \equiv |a_1\rangle\langle a_2|\,\mathrm{Tr}\Big[\,|b_1\rangle\langle b_2|\,\Big] = |a_1\rangle\langle a_2|\,\langle b_2|b_1\rangle \tag{A.35}$$

where $|a_1\rangle$ and $|a_2\rangle$ are any two vectors in the state space of A, and $|b_1\rangle$ and $|b_1\rangle$ are any two vectors in the state space of B. To fully define the partial trace, in addition to (A.35) one must also require that it acts *linearly* on its input.

Remark It is not immediately clear in what sense the reduced density operator of system A can be regarded as a description of the state of A. The physical justification lies in the fact that the reduced density operator reproduces the correct measurement statistics for all measurements performed solely on system A, as discussed in detail in [44].

A first property of reduced density operators follows directly from (A.35). Indeed,

Prop. A.2. If a quantum system is in the product state $\rho^{AB} = \rho \otimes \sigma$, where ρ is a density operator for system A, and σ is a density operator for system B; then

$$\begin{aligned} \rho_A &= \mathrm{Tr}_B[\rho\otimes\sigma] = \rho\,\mathrm{Tr}[\sigma] = \rho \\ \rho_B &= \mathrm{Tr}_A[\rho\otimes\sigma] = \sigma\,\mathrm{Tr}[\rho] = \sigma \end{aligned} \tag{A.36}$$

A less trivial, yet particularly important property (since it constitutes a key hallmark of quantum entanglement) is the following:

Prop. A.3. ρ^{AB} is entangled if and only if ρ^A is in a mixed state.

This is a remarkable result: even if the joint state of two qubits is pure (i.e., known exactly), each individual qubit is in a mixed state, about which we lack maximal knowledge. Property A.3 is difficult to prove in full generality, but a simple illustration comes from one of the four famous **Bell states** (which are all pure).

$$|\psi\rangle = \frac{1}{\sqrt{2}}\Big(\,|00\rangle + |11\rangle\,\Big) \tag{A.37}$$

Computing the reduced density operator for the subsystem A one finds

$$\rho^A = \frac{\mathbb{I}}{2} \tag{A.38}$$

which is a mixed state according to Prop. A.1, indeed

$$\mathrm{Tr}\left[\rho_A^2\right] = \frac{1}{4}\,\mathrm{Tr}[\mathbb{I}] = \frac{1}{4}\cdot 2 = \frac{1}{2} < 1 \tag{A.39}$$

A.5 States Decompositions

Density operators and the partial trace are only the starting point in the toolbox for analyzing composite quantum systems, which lie at the core of quantum computation and quantum information. Three further tools of central importance are the **Schmidt decomposition**, the **polar decomposition** and the **singular value decomposition**, which we introduce in this section, beginning with the Schmidt decomposition[7].

> **Th. A.1.** Suppose $|\psi\rangle$ is a pure state of a composite system, AB. Then there exist orthonormal states $|i_A\rangle$ for system A, and orthonormal states $|i_B\rangle$ of system B such that
>
> $$|\psi\rangle = \sum_i \lambda_i |i_A\rangle |i_B\rangle \tag{A.40}$$
>
> where λ_i are non-negative real numbers satisfying
>
> $$\sum_i \lambda_i^2 = 1 \tag{A.41}$$
>
> known as **Schmidt coefficients**.

This is a very useful result since, for pure states, one will have

$$\begin{aligned} \rho^A &= \sum_i \lambda_i^2 |i_A\rangle\langle i_A| \\ \rho^B &= \sum_i \lambda_i^2 |i_B\rangle\langle i_B| \end{aligned} \tag{A.42}$$

Thus, the eigenvalues of ρ^A and ρ^B coincide and are given by λ_i^2 for both reduced density operators. Many important characteristics of quantum systems are determined entirely by these eigenvalues, so for a pure state of a composite system such properties will necessarily be identical for both subsystems. In particular, this brings us to the following definition, directly associated with the Schmidt decomposition:

> **Def. A.3.** The bases $|i_A\rangle$ and $|i_B\rangle$ are called the **Schmidt bases** for A and B, respectively, and the number of non-zero values λ_i is called the **Schmidt number** (or **Schmidt rank**) r for the state $|\psi\rangle$.

The Schmidt number is a key property of a composite quantum system, as it provides a direct indicator of entanglement between subsystems A and B. In fact, the only states with $r = 1$ are pure product states, which by Def. A.1 are not entangled. Conversely, any entangled state necessarily has $r > 1$. Thus, the value of r already gives us a clear criterion for distinguishing separable from entangled states.

We now turn to two further decompositions: the *polar decomposition* and the *singular value decomposition*[8](SVD). Both provide powerful ways of expressing linear operators in simpler terms, in particular as products of unitary and positive operators. Since the SVD is defined in terms of the polar decomposition, we begin by introducing the latter.

[7] Proofs of all these decompositions can be found in [44].

[8] Unlike eigenvalue decompositions, the singular value decomposition exists for every real or complex matrix, making it a universally applicable tool.

Th. A.2. Let A be a linear operator on a vector space V. Then there exists unitary $\mathcal{U}$ and positive operators J and K such that

$$A = \mathcal{U}J = K\mathcal{U}^{9} \tag{A.43}$$

where the unique positive operators J and K satisfying these equations are defined by $J \equiv \sqrt{A^\dagger A}$ and $K \equiv \sqrt{AA^\dagger}$. Moreover, if A is invertible then $\mathcal{U}$ is unique.

The SVD combines the polar decomposition and the *spectral theorem.*

Th. A.3. Let A be a $m \times n$ complex matrix. Then there exist unitary matrices $\mathcal{U}$ and V, and a diagonal matrix $\mathcal{D}$ with non-negative entries such that

$$A = \mathcal{U}\mathcal{D}V^\dagger \tag{A.44}$$

The diagonal elements of $\mathcal{D}$ are called the **singular values** of A.

Moreover, the singular values are uniquely determined by A. The number of non-zero singular values equals the rank of A, and in general they are given by the square roots of the eigenvalues of $A^\dagger A$. In the special case where A is Hermitian and positive semidefinite, such as a density matrix, the singular values coincide with the eigenvalues.

We conclude this section with a useful method for relating pure and mixed states, called **purification**. Given a state ρ^A of a quantum system A, one can introduce an auxiliary system R and construct a pure state $|AR\rangle$ of the joint system such that

$$\rho^A = \mathrm{Tr}\Big[\,|AR\rangle\langle AR|\,\Big] \tag{A.45}$$

That is, the pure state $|AR\rangle$ reduces to ρ^A when restricted to system A. Since this is a purely mathematical construction, the auxiliary system R is called a *reference system*: it is fictitious and has no direct physical meaning.

As shown in [44], any mixed state can be purified using the Schmidt decomposition by defining a pure state whose Schmidt basis on A is the eigenbasis of ρ^A, with Schmidt coefficients given by the square roots of its eigenvalues.

For any given state, the purification is not unique; different purifications are related by the **Schrödinger-HJW theorem** (also known as the **purification theorem**) [138,139].

Th. A.4. Let ρ^A be a mixed quantum state, and let $|\varphi_{AR}\rangle$ and $|\psi_{AR'}\rangle$ be two purifications of ρ^A on (possibly different but equally dimensional) reference systems R and R'. Then these purifications differ by a unitary acting only on the reference systems: there exists a unitary operator $\mathcal{U}_{RR'}$ such that

$$|\psi_{AR'}\rangle = \big(\mathbb{I}_A \otimes \mathcal{U}_{RR'}\big)\,|\varphi_{AR}\rangle^{10} \tag{A.46}$$

[9] $A = \mathcal{U}J$ and $A = K\mathcal{U}$ are called *left* and *right polar decomposition*, respectively. In practice, the qualifiers “left” and “right” are often omitted, with the context making clear which form is intended.

[10] The notation $\mathbb{I}_A \otimes \mathcal{U}_{RR'}$ emphasizes that no transformation occurs on system A, while $\mathcal{U}$ acts solely on the reference systems.

A.6 Quantum Channels and Circuits

By QM's second postulate, closed systems evolve unitarily. Realistic systems, however, are rarely isolated: interactions with an external environment lead to non-unitary dynamics. Such open-system dynamics are described by **quantum channels**, which generalize unitary transformations to the most general physically admissible state evolutions. To formalize this notion, one adopts an *axiomatic approach*: a quantum channel $\mathcal{E}(\cdot)$ must satisfy two properties ensuring physical consistency, the first being **trace preservation**:

$$\mathrm{Tr}\left[\mathcal{E}(\rho)\right] = \mathrm{Tr}[\rho] \tag{A.47}$$

The second property that quantum channels must satisfy is **complete positivity**:

Def. A.4. A linear map $\mathcal{E}$ from operators on a Hilbert space $\mathcal{H}_A$ to operators on $\mathcal{H}_B$ is completely positive if, for every auxiliary Hilbert space $\mathcal{H}_R$ of arbitrary dimension, the extended map

$$\mathbb{I}_R \otimes \mathcal{E} : \mathcal{L}(\mathcal{H}_R \otimes \mathcal{H}_A) \to \mathcal{L}(\mathcal{H}_R \otimes \mathcal{H}_B) \tag{A.48}$$

maps positive operators to positive operators. In particular,

$$A \geq 0 \quad \Longrightarrow \quad (\mathbb{I}_R \otimes \mathcal{E})(A) \geq 0 \tag{A.49}$$

for all $A \in \mathcal{L}(\mathcal{H}_R \otimes \mathcal{H}_A)$[11].

Together, these two properties guarantee physical consistency. Complete positivity is stronger than positivity alone, since every completely positive map is positive, but not vice versa. Physically, it ensures that a channel remains valid even when the system is entangled with an arbitrary ancilla: the extended map $\mathbb{I} \otimes \mathcal{E}$ must still yield valid density operators. With these requirements in place, we can now define a *quantum channel*:

Def. A.5. A quantum channel is a linear map on density operators that is both trace preserving and completely positive (CPTP).

The equivalence between this axiomatic definition of quantum channels their characterization through the operator-sum representation is established by **Choi's theorem**[12]:

Th. A.5. The map $\mathcal{E}$ is a quantum channel if and only if

$$\mathcal{E}(\rho) = \sum_i K_i \, \rho \, K_i^{\dagger} \text{[13]} \tag{A.50}$$

for some set of operators $\{K_i\} : \mathcal{H}_{in} \to \mathcal{H}_{out}$ which

$$\sum_i K_i \, K_i^{\dagger} = \mathbb{I} \tag{A.51}$$

[11] We denote with $\mathcal{L}(\mathcal{H})$ the set of all linear operators acting on the Hilbert space $\mathcal{H}$.

[12] The proof can be found in [44].

[13] Eq. (A.50) is known as **Kraus representation** and the K_i are called **Kraus operators**.

Although quantum channels are not necessarily unitary, there is a fundamental result, proven in [44], showing that any such map can always be *realized physically* as a unitary transformation on an enlarged Hilbert space.

Th. A.6. Let $\mathcal{E} : \mathcal{L}(\mathcal{H}_A) \to \mathcal{L}(\mathcal{H}_A)$ be a quantum channel. Then there exist an auxiliary Hilbert space $\mathcal{H}_E$[14], a state $|e_0\rangle \in \mathcal{H}_E$, and a unitary operator

$$\mathcal{U}_{AE} : \mathcal{H}_A \otimes \mathcal{H}_E \longrightarrow \mathcal{H}_A \otimes \mathcal{H}_E \tag{A.52}$$

such that

$$\mathcal{E}(\rho) = \mathrm{Tr}_E \left[\mathcal{U}_{AE} \big(\rho_A \otimes |e_0\rangle\langle e_0| \big) \mathcal{U}_{AE}^\dagger \right]. \tag{A.53}$$

This construction is known as the **Stinespring representation** of $\mathcal{E}$.

Unitary quantum channels are commonly referred to as **quantum gates**, and, apart from measurements, they are the only operations allowed in the standard *quantum circuit model*. More general quantum channels are frequently employed to describe **noise** processes, such as the *bit-flip*, *phase-flip*, *ampltude-damping* or *depolarizing channels*, which capture the effects of decoherence and errors in realistic devices.

A **quantum circuit** is then defined as a sequence of unitary operations applied to an initial set of qubits; these gates can act on single qubits or on pairs of qubits.

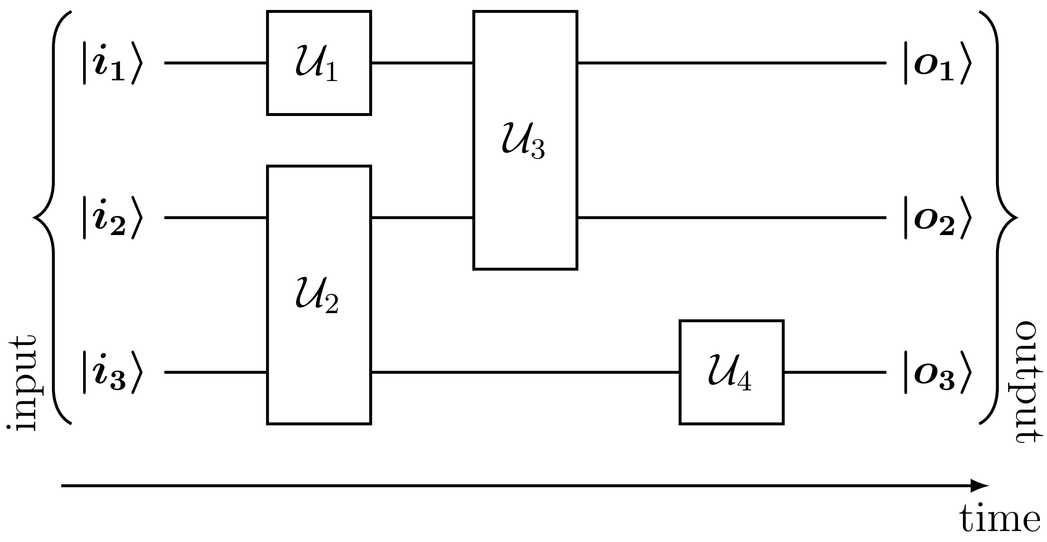


Figure A.4: *Schematic representation of a quantum circuit: input qubits are, unless otherwise specified, initialized in the $|0\rangle$ state and evolve through a sequence of unitary operations into output states. In this notation, time always flows from left to right. The output qubits are eventually measured, although the measurement symbol is not shown here and will be introduced later.*

[14] The subscript E denotes the environment associated with system A.

We now introduce the most commonly used quantum gates, beginning with single-qubit gates, which are easy to visualize as rotations on the Bloch sphere (A.26). The first set is given by the Pauli gates, which directly implement the Pauli matrices (A.27). Each Pauli gate corresponds to a rotation by π around its associated axis on the Bloch sphere: X around the x-axis, Y around the y-axis, and Z around the z-axis.

Beyond the Pauli gates, other fundamental single-qubit gates are the **Hadamard** gate H, which performs a rotation of π around the axis $(x+z)/\sqrt{2}$ and plays the key role of creating *superposition states*, the **phase** gate S, corresponding to a rotation of $\pi/2$ around the z-axis, and the $\boldsymbol{T}$ gate, corresponding to a rotation of $\pi/4$ around the z-axis. Their action is fully specified by the following matrix representations:

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{A.54}$$

$$S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} \tag{A.55}$$

$$T = \begin{pmatrix} 1 & 0 \\ 0 & \exp\left[i\frac{\pi}{4}\right] \end{pmatrix} \tag{A.56}$$

Moving on to two-qubit gates, the most important examples are the **SWAP** gate, which exchanges the states of two qubits, the **controlled-NOT** (CNOT) gate, which flips the target qubit conditional on the control and, crucially, enables the creation of entanglement between qubits, and the **controlled-$\boldsymbol{Z}$** (CZ) gate, which applies a phase flip to the target when the control is in $|1\rangle$. Their action is given by the following matrices:

$$\text{SWAP} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{A.57}$$

$$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{A.58}$$

$$\text{CZ} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \tag{A.59}$$

Any unitary matrix acting on qubits defines a valid quantum gate. However, multi-qubit unitaries are often difficult or impractical to implement directly. Fortunately, any quantum computation can be constructed from a **finite universal gate set**, since every n-qubit unitary can be decomposed into one- and two-qubit gates. A widely used (though not unique) choice of universal set is

$$\{\text{CNOT}, H, S, T\} \tag{A.60}$$

It can be shown that any unitary operator $\mathcal{U} \in U(N)$ can be *expressed exactly* as a finite sequence of single-qubit operations and CNOT gates. This result assumes access to an ideal, *continuous* set of single-qubit rotations, allowing arbitrary parameters.

In practice, however, physical implementations rely on a *discrete* set of gates. Remarkably, it can be proven that any single-qubit operation can be *approximated to arbitrary accuracy* ϵ using only gates from the universal set (A.60).

In summary, *exact universality* is achieved when arbitrary continuous single-qubit rotations are available, while *approximate universality* arises when one restricts to a finite, discrete set of gates that densely generates $U(N)$.

For clarity, the main circuit symbols are collected in the following figure.

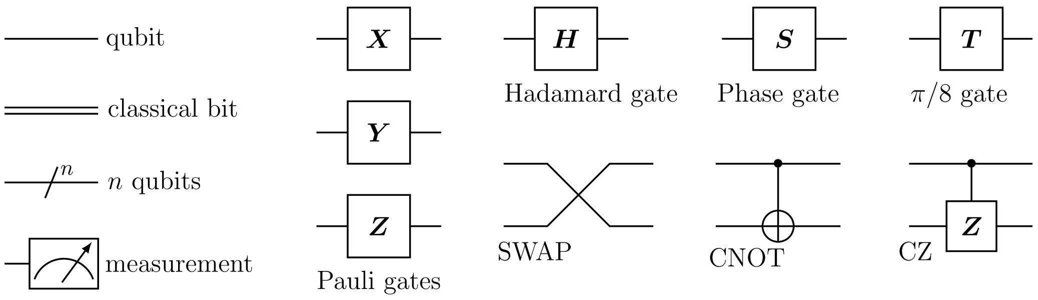


Figure A.5: *Circuit symbols for common quantum operations: single-qubit gates (X, Y, Z, H, S, T), two-qubit gates (SWAP, CNOT, CZ), measurement, and notations for qubits, classical bits, and multi-qubit registers.*

We close this section by reporting an important result: any classical logic circuit can be simulated using a quantum circuit. The equivalence is not straightforward though, since quantum logic gates are inherently reversible (since they are unitary by definition), whereas many classical logic gates such as the `NAND` gate are inherently irreversible.

Another major difference between classical and quantum computation is the *impossibility of cloning* an arbitrary quantum state, known as the **no-cloning theorem** [140]. While a classical bit can be copied using a simple CNOT gate, an unknown quantum state cannot be duplicated. Intuitively, this reflects the fact that a qubit contains information that is not directly accessible through measurement. For instance, in the entangled state $a\,|00\rangle + b\,|11\rangle$, measuring one qubit yields 0 or 1 with probabilities $|a|^2$ and $|b|^2$, and simultaneously determines the state of the other qubit. The coefficients a and b, the "hidden" information of the original state $|\psi\rangle$, are lost upon measurement and cannot be recovered. If perfect copying were possible, that information would persist in the clone, contradicting quantum mechanics [44].

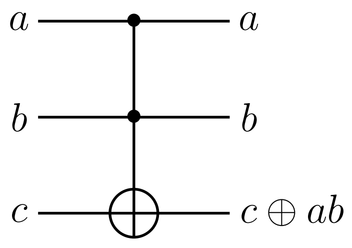
Fortunately, any classical circuit can be transformed into an equivalent one built entirely from reversible elements, using the **Toffoli gate** as a fundamental building block. Its circuit symbol and matrix representation are shown in Fig. A.6. Since the Toffoli gate admits a direct quantum implementation, it follows that, in principle, *every* classical circuit can be simulated by a quantum circuit.

We conclude this section by noting that quantum computers can also efficiently simulate non-deterministic classical computers, i.e., machines capable of generating random bits.

To achieve this, it suffices to produce fair coin tosses, which can be realized by preparing a qubit in $|0\rangle$ (or $|1\rangle$), applying a Hadamard gate (A.54) to obtain the *superposition states*

$$|\pm\rangle = \frac{|0\rangle \pm |1\rangle}{\sqrt{2}}, \tag{A.61}$$

and then measuring the state. The outcome will be $|0\rangle$ or $|1\rangle$, with equal probability.



$$\Longrightarrow \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Figure A.6: *Circuit symbol and matrix representation of the Toffoli (CCNOT) gate. The target qubit is flipped when both control qubits are in $|1\rangle$. Here, $\oplus$ denotes addition modulo 2.*

B

Complexity Theory

B.1 General Overview

Computational complexity theory (or simply *complexity theory*) is the study of the minimal resources required to solve computational problems. In particular, it aims to distinguish between problems that admit efficient algorithms (the "easy" problems) and those that are inherently intractable (the "hard" problems). Computational complexity also provides the foundations for most of modern *cryptography*, whose goal is to design cryptosystems that are "easy to use" but "hard to break" [141].

In the most general terms, a *computational problem* can be defined as in [9]: a problem is a general question to be answered, typically involving several parameters (or free variables) whose values are left unspecified. A problem is described by providing a general description of all its parameters and a statement of the properties that the answer, or solution, is required to satisfy. An *instance* of a problem is obtained by assigning specific values to all the problem's parameters. Within this general framework, algorithms can be defined as general, step-by-step procedures for solving problems. An algorithm is said to *solve* a given problem Π if it can be applied to any instance I of Π and is guaranteed always to produce a solution for that instance.

Once an algorithm is found, how do we "quantify" its efficiency in solving Π? The most basic resource studied in computational complexity is the *running time*, defined as the number of "*elementary operations*" performed by an algorithm. However, providing a formal definition of an elementary operation can be tricky; therefore, to make it precise, one must fix a *model of computation* (such as the *Turing machine* [142]). Typically, the running time is measured as a *function of the input length*. For the vast majority of computational problems, the input is expressed as a finite sequence of bits, i.e., a finite sequence of zeroes and ones. The *set of all strings* (see Def. B.2 below) *of length* n is denoted by $\{0,1\}^n$, while the *set of all strings* is denoted by $\{0,1\}^* = \cup_{n\geq 0}\{0,1\}^n$.

To express more formally the concept of an algorithm's "efficiency," we introduce the so-called **Big-Oh notation** (or simply **O-notation**) [136,143], which is useful for expressing asymptotic upper bounds of functions.

> **Def. B.1.** If f and g are two functions from $\mathbb{N}$ to $\mathbb{N}$, we say that $f = O(g)$ if there exist positive constants c and n_0 such that $0 \leq f(n) \leq c\,g(n),\ \forall n \geq n_0$.

In this way, we can express the efficiency of an algorithm using a function f from the set of natural numbers $\mathbb{N}$ to itself, where $f(n)$ denotes the maximum number of basic operations the algorithm performs on inputs of length n. If the algorithm runs in **polynomial time**, namely $f(n) = O(n^c)$ for some constant c, we say that it is **efficient**.

Now, to distinguish among different computational problems, we need the concept of **complexity classes**; however, to define them, we must first fix a computational model. Since, historically, they were first defined in terms of Turing machines, we will proceed in the same way and introduce this model in the following section.

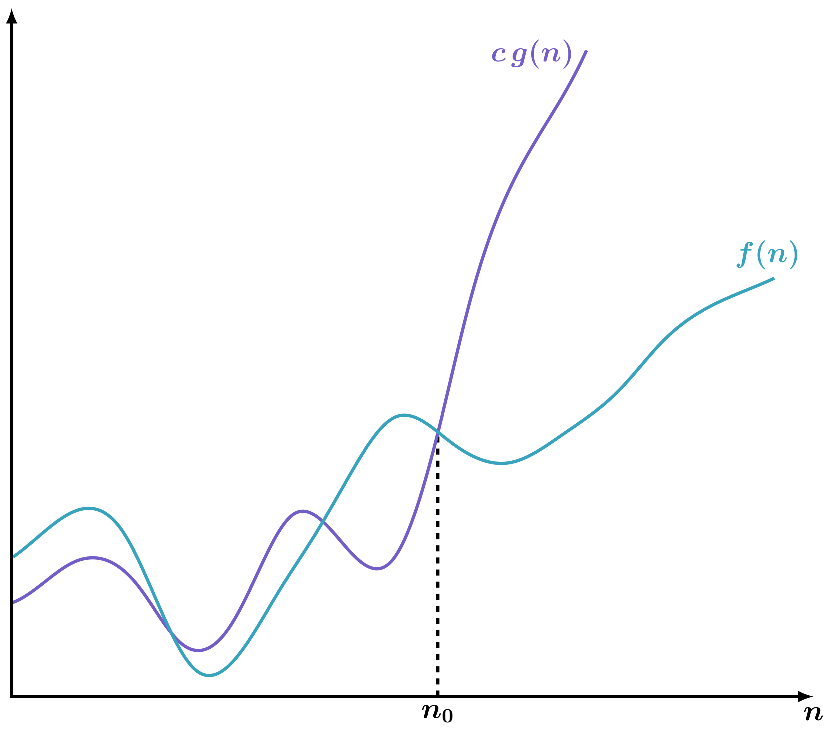


Figure B.1: *Graphic example of the O-notation B.1, where we have two positive constants c and n_0 such that at and to the right of n_0, the value of $f(n)$ always lies on or below $c\,g(n)$.*

B.2 Turing Machines

Before introducing the Turing machines' model, it's useful to formalize the basic objects they manipulate, that is **alphabets** and **strings** [144].

> **Def. B.2.** An *alphabet* Σ is a finite non-empty set, whose elements are called letters or symbols. A *string* over Σ is a finite sequence of symbols in Σ.

Turing machines are a class of "*automatic machines*" introduced by Turing In 1936, with the aim of providing a rigorous formulation of the notion of an algorithm for executing a computational task. A Turing machine constitutes a mathematical model of computation, representing an abstract device that processes symbols on an infinite tape according to a finite set of deterministic transition rules. In particular, following [44]:

> **Def. B.3.** A *Turing machine* (TM) is a mathematical model of computation describing an abstract machine that manipulates symbols from an alphabet Σ on a strip of tape according to a table of rules with the goal to compute the value of a function on a given input. A TM is made of four main elements:
>
> - A *finite-state-control*
> - A *memory tape*
> - A *read-write tape-head*
> - A *program*

We now examine each of these elements in detail to clarify their roles:

- The finite-state-control consists of a finite set of *internal states* $q_1, \dots q_m$ plus two special internal states, labelled q_s (*starting state*) and q_h (*halting state*).
- The tape is a one-dimensional object, which stretches off to infinity in one direction, consisting of an infinite sequence of *tape squares* each containing one symbol drawn from some alphabet Σ which, for simplicity, we assume to contain just four symbols: 0, 1, b (the "blank" symbol) and $\triangleright$ (to mark the left hand edge of the tape).
- The read-write tape-head identifies a single square on the tape as the square that is currently being accessed by the machine.
- A program for a Turing machine is a finite, ordered list of instruction lines of the form $\langle q, x, q', x', s \rangle$ where q and q' are internal states, $x, x' \in \Sigma$, and $s \in \{0, \pm 1\}$. If the current internal state is q and the symbol under the head is x, the machine searches for an instruction with matching q and x. When such an instruction is found, it is *executed*: the state changes to q', the symbol x is overwritten by x', and the head moves left, right, or remains stationary depending on whether s equals -1, $+1$, or 0, respectively. If no matching instruction exists, the internal state of the machine is changed to q_h.

Summarizing, the machine starts its operation with the finite state control in the state q_s, and with the read-write head at the leftmost tape square. The computation then proceeds in a step by step manner according to the program. If the current state is q_h, then the computation has halted, and the output of the computation is the current (non-blank) contents of the tape. Fig. B.2 gives a schematic representation of a TM.

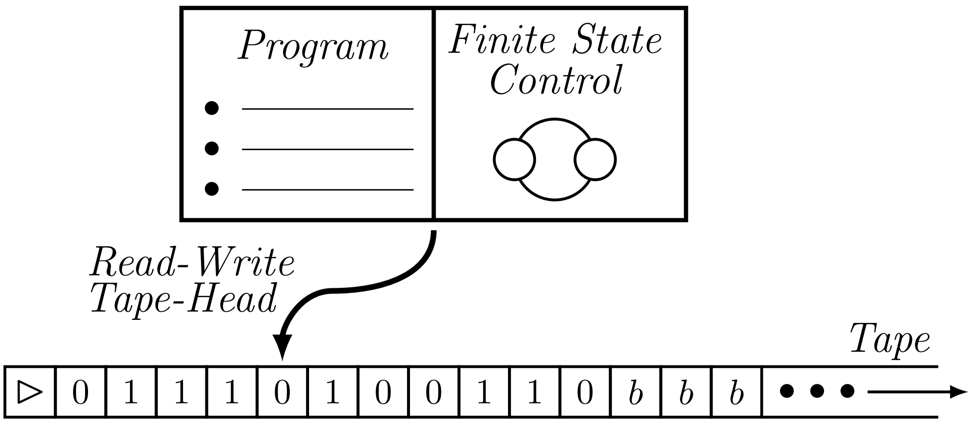


Figure B.2: *Main elements of a Turing machine.*

Simultaneously with Turing's work, Alonzo Church introduced a method for defining functions known as the λ-*calculus* [145], and called λ-*computable* any function on the natural numbers that can be represented by a λ-calculus term. What both Turing [146] and Church [147] later proved is that the notions of λ-computable and *Turing-computable* functions (where a function is Turing-computable if it can be computed by a TM) are equivalent. Moreover, they also showed that these classes coincide with the class of μ-*recursive* (or *general recursive*) functions, originally proposed by Gödel and Herbrand in 1934 [148] and later formalized and published by Kleene in 1936 [149].

All these equivalences are encapsulated in the well-known **Church-Turing thesis**:

> *The class of functions computable by a TM corresponds exactly to the class of functions which we would naturally regard as being computable by an algorithm.*

Equivalently, it can be stated more informally as follows [150].

> *A function is effectively computable, that is, computable by means of a finite, well-defined procedure such as calculations performed with paper and pencil, if and only if it is Turing-computable.*

It may appear obvious that every function can be computed given enough time. Yet some functions cannot be computed in any finite number of steps. Formally, this is captured by the following theorem, whose proof is given in [143].

Th. B.1. $\exists\, f : \{0,1\}^* \to \{0,1\}$ that isn't computable by any TM.

There are many examples of *uncomputable* (also called *undecidable*) functions, the most famous being the so-called `HALT` function. This function takes as input a pair (α, x) and outputs 1 if and only if the TM represented by the *Turing number*, hat is, a natural number encoding the machine in a standard enumeration of all Turing machines, uniquely identifying it up to the choice of encoding, M_α halts on input x within a finite number of steps. It turns out that this function is not Turing-computable, as stated in the following theorem, whose proof can again be found in [143].

Th. B.2. `HALT` is not computable by any TM.

A priori, it is not evident that every function we would intuitively regard as computable by an algorithm can, in fact, be computed by a TM. Church, Turing, and many others have devoted substantial effort to collecting evidence in support of the Church-Turing thesis, and to date no counterexamples have been found. Nevertheless, the possibility remains that a function violating the thesis could exist. The discovery of such a function would force a radical revision of whole areas of logic and theoretical computer science, most notably complexity theory, requiring a reformulation from the ground up and revealing the incompleteness of Turing's model as encapsulated in the Church-Turing thesis.

There are many **variants** of the basic TM model. For example, one may consider machines with different types of tapes, such as two-way infinite tapes or multidimensional tapes. As far as is currently known, no modification of the Turing model that is physically reasonable leads to a strictly larger class of computable functions.

This invariance of computational power under reasonable changes to the definition is usually referred to as **robustness**. In particular, it can be shown that all such variants of Turing machines are *equivalent*, meaning that each model can simulate the other. For instance, the equivalence between the multitape and the single-tape models is established by the following theorem, whose proof is given in [151].

Th. B.3. Every multitape TM has an equivalent single-tape TM.

Another important variant of Turing's model is the **nondeterministic Turing machine** (NTM), which, at any point in the computation, may proceed according to several possible transitions. Formally, such machines are governed by transition functions of the form

$$\delta : Q \times \Sigma \longrightarrow Q \times \Sigma \times \{L, R\} \tag{B.1}$$

where Q is the set of internal states, Σ the tape alphabet, and $\{L, R\}$ specifies the head movement (Left or Right). In other words, for each pair of current state and tape symbol, the transition function returns a set of possible instructions rather than a single one. The computation of an NTM can thus be visualized as a tree whose branches represent the different possible choices at each step. An input is accepted if at least one computation branch leads to an accepting state. Despite this apparent additional flexibility, nondeterminism does not affect the power of the Turing machine model, as established by the following theorem, whose proof is again in [151].

Th. B.4. Every nondeterministic TM has an equivalent deterministic TM.

Among the different NTMs, the most significant is the **probabilistic TM** [143, 151].

Def. B.4. A *probabilistic Turing machine* (PTM) M is a nondeterministic TM in which each nondeterministic step is interpreted as a *coin-flip step*. At such a step, the machine has exactly two possible successor configurations, corresponding to applying one of two transition functions, δ_1 or δ_2, both of the form given in (B.1). Each computation of M on input x can be represented as a branch in the computation tree. We assign a probability to each branch b by defining

$$\Pr[b] = 2^{-k}, \tag{B.2}$$

where k denotes the number of coin-flip steps along branch b. The probability that M accepts x is then defined as

$$\Pr[M \text{ accepts } x] = \sum_{b \in \mathcal{B}_{\text{acc}}} \Pr[b], \tag{B.3}$$

where $\mathcal{B}_{\text{acc}}$ denotes the set of all accepting branches of the computation tree.

It is important to emphasize that each coin-flip step corresponds to an independent random choice, with probability $1/2$ of applying δ_1 and probability $1/2$ of applying δ_2; moreover, the machine does not retain any information about previous outcomes beyond what is already encoded in its current configuration [143].

PTMs are of particular importance in light of the **Strong Church-Turing thesis** [44], since they capture the idea that allowing randomness may yield more *efficient* algorithms than those obtainable with purely deterministic Turing machines.

Any model of computation can be simulated on a PTM with at most a polynomial increase in the number of elementary operations required.

The Strong Church-Turing thesis essentially states that the entire theory of computational complexity can be framed in terms of PTMs: if a problem admits a polynomial-time algorithm on some model of computation, then it also admits one on a PTM; conversely, if no such algorithm exists on a PTM, then no efficient algorithm exists at all.

B.3 Complexity Classes

One of the central goals of complexity theory is to classify computational problems according to their intrinsic difficulty: given a problem, how much computing power and how many resources are needed to solve it? Although precise answers to such questions remain out of reach, substantial progress has been made in grouping problems into complexity classes, which capture, at least approximately, their inherent computational difficulty. In what follows, we introduce the main classes, explain how problems can be categorized according to the time required to solve them, and discuss what is known about the relationships between these classes. Before proceeding, we introduce some preliminary notions. The most fundamental of these is the concept of a **decision problem** [152], since most computational problems can be formulated in this way.

> **Def. B.5.** A *decision problem* is a computational problem whose answer is restricted to either "*yes*" or "*no*". Formally, it can be represented as a Boolean function $f : \{0,1\}^* \longrightarrow \{0,1\}$ where the input is a binary string of arbitrary length and the output is 1 (for "*yes*") or 0 (for "*no*").

Equivalently, decision problems are often described in terms of **languages** [44]

> **Def. B.6.** A *language* L over the alphabet Σ is a subset of the set Σ^* of all (finite) strings of symbols from Σ[1].

Thus, every decision problem can be equivalently expressed as the task of deciding whether a given string is an element of the language L. With these preliminaries in place, we can now formally introduce the concept of a **complexity class** [152]

> **Def. B.7.** A *complexity class* is a set of decision problems (i.e., languages) that all have similar computational complexity

As anticipated, the most commonly used resource for quantifying the complexity of an algorithm is its *running time* (corresponding to the number of basic operations performed). It is therefore natural to introduce the notion of a **time complexity class** [151].

> **Def. B.8.** Let $t : \mathbb{N} \longrightarrow \mathbb{N}$ be a function. Define the *time complexity class*, $\mathrm{TIME}(t(n))$, to be the collection of all languages that are decidable by an $O(t(n))$ running-time TM.

Having established the necessary preliminaries, we can now proceed to the definition of the major complexity classes, starting with the class **P** [151].

> **Def. B.9.** P is the class of languages that are *decidable* in polynomial time on a deterministic TM. In other words,
>
> $$P = \bigcup_k \mathrm{TIME}\big(n^k\big) \tag{B.4}$$

[1] In most practical cases $\Sigma \equiv \{0,1\}$ and thus a language will just be any subset of $\{0,1\}^*$ [153].

Next, we move to the class **NP** [152,153], which plays a central role in complexity theory since a vast number of fundamental computational problems lie within it.

> **Def. B.10.** *NP* is the class of languages L for which *membership* (that is, inputs for which the correct answer is ”yes”) can be *verified* in polynomial time by a deterministic TM, given an appropriate certificate. Equivalently, *NP* is the class of languages *decidable* in polynomial time on a nondeterministic TM.

Not all problems within the same complexity class are equally difficult. To compare their relative difficulty and establish a hierarchy among them, we first introduce the notions of **reducible** languages and **reductions** [151].

> **Def. B.11.** Language A is said to be (mapping) *reducible* to language B if there is a computable function $f : \Sigma^* \longrightarrow \Sigma^*$, where for every x,
>
> $$x \in A \iff f(x) \in B \tag{B.5}$$
>
> The function f is called the *reduction* from A to B. If f can be computed in polynomial time, the reduction is called a *polynomial-time reduction.*

It is important to note that if a language A is reducible to a language B, this implies that A is *no harder than* B: an efficient algorithm for B immediately yields an efficient algorithm for A. However, this does not mean that A and B are equally difficult. Only when reductions exist in both directions ($A \leq_r B$ and $B \leq_r A$) the two problems can be regarded as *equivalent in complexity.* Having introduced the notion of reductions, we can now formally define the concept of **complete** languages [151].

> **Def. B.12.** Let $\mathcal{C}$ be a complexity class. A language A is said to be $\mathcal{C}$*-complete* if it satisfies the two following conditions:
>
> - $A \in \mathcal{C}$
> - Every $B \in \mathcal{C}$ is polynomial-time reducible to A (i.e., A is $\mathcal{C}$*-hard*)

It is worth stressing that the definition of $\mathcal{C}$-hardness does not require the problem itself to lie within the class $\mathcal{C}$ (for instance, the *Halting Problem* is NP-hard but not in NP). By contrast, a $\mathcal{C}$-complete problem is both in $\mathcal{C}$ and $\mathcal{C}$-hard, and thus represents the “hardest” problems of the class. Consequently, if an efficient algorithm were found for any $\mathcal{C}$-complete problem, every problem in $\mathcal{C}$ could be solved efficiently as well.

Having given the definition of $\mathcal{C}$-completeness, we can now introduce the well-known **P vs NP problem**. This question was first formulated in the early 1970s, when Cook [7] and Levin [154,155] independently proved that the *Boolean satisfiability problem* (SAT) is **NP-complete**, a result that has since become known as the *Cook–Levin Theorem.*

By definition it is clear that $P \subseteq NP$, since every problem that can be solved in polynomial time can also have its solutions verified in polynomial time. What remains unknown, however, is whether the converse inclusion holds: does $NP \subseteq P$? In other words, is every efficiently verifiable problem also efficiently solvable? Resolving this question, whether $P = NP$ or $P \neq NP$, still remains one of the deepest open problems in theoretical computer science [156,157], whose resolution would fundamentally reshape our understanding of computer science and computational complexity.

In addition to running time, another fundamental computational resource is the *space* used by a TM, that is, the number of tape cells it scans during the computation. Considering space as the complexity measure leads to the definition of the class **PSPACE** [151].

> **Def. B.13.** $PSPACE$ is the class of languages that are decidable in polynomial space on a deterministic TM.

By their very definitions, we immediately obtain the chain of inclusions $P \subseteq NP \subseteq PSPACE$. Indeed, every nondeterministic polynomial-time computation uses at most a polynomial number of tape cells, so $NP \subseteq PSPACE$. While these containments are straightforward, what remains unknown is whether they are strict. In particular, the question of whether $NP = PSPACE$ or $NP \subsetneq PSPACE$ is still an open problem [158].

The last notable classical complexity class to be introduced is the widely studied **BPP**, which stands for Bounded-Error Probabilistic Polynomial time [151].

> **Def. B.14.** BPP is the class of languages that are decidable in polynomial time on a PTM with an error probability of $1/3$.

Again, by the definitions, we have the inclusions $P \subseteq BPP \subseteq PSPACE$, where the first containment holds because every deterministic polynomial-time algorithm is a special case of a probabilistic one, while the second follows from the fact that a $PSPACE$ machine can simulate all possible random choices using only polynomial space.

It is widely conjectured that randomness does not increase computational power asymptotically, leading to the belief that $P = BPP$. However, the precise relationship between the BPP and NP classes remains unknown: it is still not established whether $BPP \subseteq NP$, $NP \subseteq BPP$, or neither [159].

The only quantum complexity class we (informally) introduce is the analogue of BPP, namely **BQP**, which stands for Bounded-Error Quantum Polynomial time [44].

> **Def. B.15.** BQP is the class of computational problems that can be solved efficiently on a quantum computer, with an error probability of $1/3$.

It is important to note that the error bound of $1/3$ which appears in the definitions of both BPP (Def. B.14) and BQP (Def. B.15) is arbitrary, though it is the most common convention. In fact, any constant error probability strictly between 0 and $1/2$ leads to an equivalent definition of the class, as shown in [151]. As an example, a less common choice for ϵ is $1/4$, as adopted in [160, 161].

From the definitions we obtain the inclusions $P \subseteq BPP \subseteq BQP \subseteq PSPACE$, where the first two follow because every deterministic or probabilistic polynomial-time algorithm can be simulated by a quantum computer, while the last inclusion holds since a PSPACE machine can simulate the evolution of a polynomial-size quantum circuit using only polynomial space. The precise relation between BQP and NP, remains unknown: it is not established whether $BQP \subseteq NP$, $NP \subseteq BQP$, or neither[2] [162, 163].

[2] It is known that BQP contains problems outside P, such as integer factoring, but to date no NP-complete problem has been proven to be in BQP.

Summarizing, the following chain of inclusions represents the only relations between these complexity classes that are formally established:

$$P \subseteq NP \subseteq PSPACE \tag{B.6}$$

$$P \subseteq BPP \subseteq PSPACE \tag{B.7}$$

$$P \subseteq BPP \subseteq BQP \subseteq PSPACE \tag{B.8}$$

Fig. B.3 illustrates schematically the possible relations between the complexity classes.

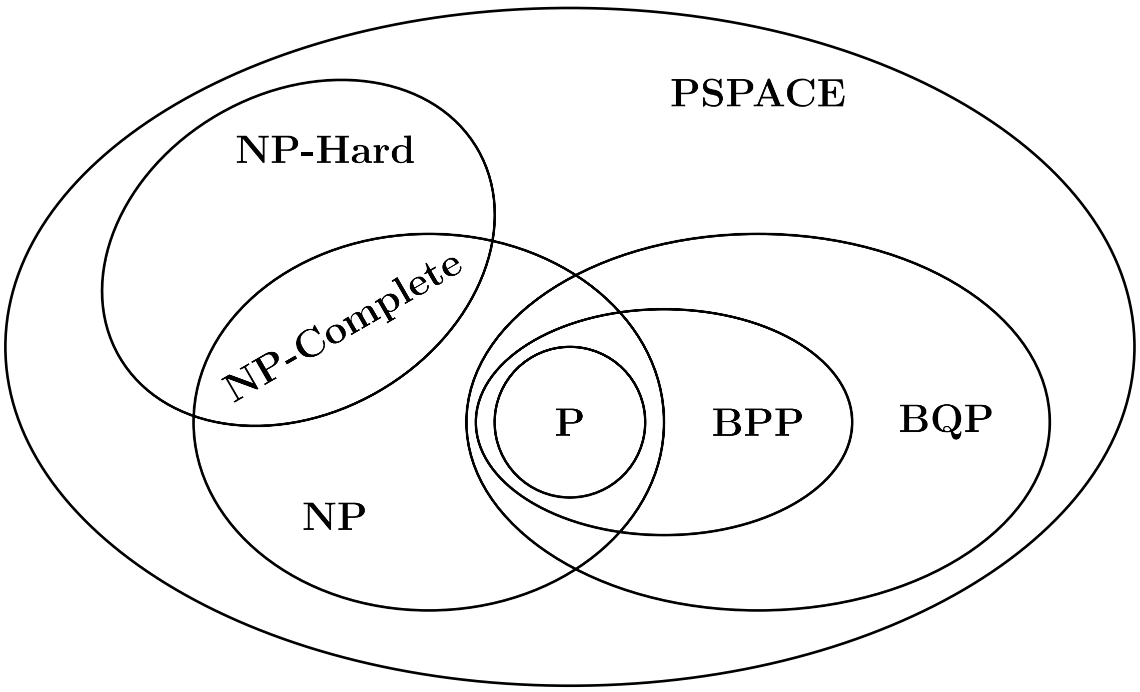


Figure B.3: *Schematic representation of the main possible relations between the complexity classes discussed. The inclusions in eqs.* (B.6) *to* (B.8) *are known, while the relations between NP and BPP/BQP remain unresolved. The classes NP-Complete and NP-Hard are shown relative to NP only.*

B.4 NP-Complete Problems

As already noted, the Boolean satisfiability problem (SAT) was the first to be shown NP-complete, a result obtained by Cook in 1971 and, independently, by Levin in 1973. Building on this foundation, in 1972 Richard Karp [8] identified 21 additional problems as NP-complete by constructing polynomial-time reductions from SAT.

The rapid development of the field culminated in 1979 with the publication of Garey and Johnson's monograph [9], which systematically presented hundreds of NP-complete problems, introduced a standardized reduction methodology, and provided a comprehensive catalog that remains a standard reference to this day.

Since then, the study of NP-complete problems has become a central theme in theoretical computer science, not only because of its foundational role in the P versus NP question[3], but also because of its practical implications: it shows that many natural problems in optimization, scheduling, graph theory, and beyond share a common barrier to efficient exact solution. For this reason, before turning to the specific problem studied in this work, it is useful to recall the general framework of NP-completeness.

As mentioned, NP-complete problems represent the most difficult problems within NP. Thus, if a single NP-complete problem were shown to admit a polynomial-time algorithm, then all problems in NP would, yielding the $P = NP$ collapse. Conversely, once a problem is established as NP-complete, this serves as evidence that no efficient exact algorithm is likely to exist, and the focus shifts toward approximation schemes, parameterized methods, or heuristic strategies, an approach that will also be adopted in this work.

In light of this intrinsic hardness, research has focused on alternative strategies rather than seeking general polynomial-time algorithms. For example, *exact exponential-time algorithms* [164] have been refined with advanced techniques to achieve running times much faster than naive enumeration. *Parameterized approaches* [165] on the other hand exploit structural properties of the input to obtain fixed-parameter tractability in restricted cases. *Approximation algorithms* [166] provide performance guarantees for some optimization variants, while *heuristics and metaheuristics*[4] [167] deliver effective solutions in practice without worst-case guarantees. Thus, although NP-complete problems are unlikely to admit efficient exact algorithms, a broad algorithmic toolbox has emerged to tackle them from both theoretical and practical perspectives.

In summary, NP-complete problems provide a unifying framework for understanding computational intractability and motivate the wide range of algorithmic techniques developed to cope with them. The problem considered in this work fall precisely within this context: while the decision version of the *Maximum Independent Set problem* (MIS) is NP-complete, we focus on the *optimization version*, which is NP-hard. This observation justifies the discussion above and clarifies why efficient exact algorithms are unlikely to exist.

[3] An efficient algorithm for any NP-complete problem would imply $P = NP$

[4] By heuristics we mean problem-specific strategies such as greedy methods or local search, while metaheuristics refer to more general optimization frameworks like simulated annealing.

C Embedding Results

r	p	$\bar{d}^S_{2D}$	$\bar{d}^F_{2D}$	$\bar{N}'$	$\bar{N}_W$	$\bar{N}_A$	$\bar{t}$ (s)
0.6	1.000	0.17	-	8.009	0.006	0.003	0.0015
0.7	1.000	0.30	-	8.041	0.030	0.011	0.0009
0.8	0.998	0.45	0.75	8.108	0.076	0.032	0.0009
0.9	0.998	0.64	1.38	8.282	0.198	0.083	0.0009
1.0	0.995	0.84	1.60	8.574	0.406	0.168	0.0010
1.1	0.984	1.08	1.86	9.201	0.898	0.303	0.0012
1.2	0.960	1.36	2.21	10.356	1.765	0.592	0.0013
1.3	0.893	1.60	2.43	11.582	2.658	0.924	0.0016
1.4	0.758	1.81	2.61	13.231	3.992	1.239	0.0027

Table C.1: *Results of the embedding algorithm for* $N = 8$.

r	p	$\bar{d}^S_{2D}$	$\bar{d}^F_{2D}$	$\bar{N}'$	$\bar{N}_W$	$\bar{N}_A$	$\bar{t}$ (s)
0.6	1.000	0.24	-	10.012	0.008	0.004	0.0019
0.7	0.998	0.42	0.80	10.110	0.090	0.020	0.0019
0.8	0.991	0.63	0.98	10.311	0.230	0.082	0.0014
0.9	0.986	0.87	1.46	10.755	0.542	0.213	0.0020
1.0	0.967	1.15	1.67	11.525	1.113	0.413	0.0023
1.1	0.918	1.46	2.10	12.963	2.227	0.736	0.0024
1.2	0.805	1.76	2.38	15.805	4.489	1.316	0.0038
1.3	0.608	2.02	2.66	18.765	6.806	1.959	0.0060
1.4	0.375	2.23	2.94	21.464	8.912	2.552	0.0059

Table C.2: *Results of the embedding algorithm for* $N = 10$.

r	p	$\bar{d}^S_{2D}$	$\bar{d}^F_{2D}$	$\bar{N}'$	$\bar{N}_W$	$\bar{N}_A$	$\bar{t}$ (s)
0.6	0.996	0.48	0.82	14.221	0.167	0.054	0.0024
0.7	0.991	0.78	1.19	14.771	0.579	0.192	0.0019
0.8	0.967	1.12	1.54	16.055	1.510	0.545	0.0023
0.9	0.876	1.47	1.96	18.386	3.272	1.114	0.0032
1.0	0.672	1.83	2.30	22.321	6.330	1.991	0.0053
1.1	0.338	2.12	2.67	27.476	10.586	2.891	0.0095
1.2	0.111	2.36	3.09	32.811	15.027	3.784	0.0180
1.3	0.020	2.55	3.56	38.850	19.900	4.950	0.0192
1.4	0.002	3.07	4.11	52.000	31.000	7.000	0.0446

Table C.3: *Results of the embedding algorithm for* $N = 14$.

r	p	$\bar{d}^S_{2D}$	$\bar{d}^F_{2D}$	$\bar{N}'$	$\bar{N}_W$	$\bar{N}_A$	$\bar{t}$ (s)
0.6	0.995	0.59	1.05	16.399	0.297	0.102	0.0029
0.7	0.973	0.95	1.31	17.447	1.091	0.356	0.0046
0.8	0.913	1.34	1.76	19.760	2.796	0.965	0.0080
0.9	0.679	1.72	2.14	23.680	5.806	1.875	0.0142
1.0	0.364	2.10	2.57	30.063	10.984	3.080	0.0267
1.1	0.089	2.37	3.03	38.213	17.820	4.293	0.0598
1.2	0.008	2.56	3.58	40.625	19.750	4.875	0.0594
1.3	0.000	-	4.18	-	-	-	-
1.4	0.000	-	4.82	-	-	-	-

Table C.4: *Results of the embedding algorithm for* $N = 16$.

r	p	$\bar{d}^S_{2D}$	$\bar{d}^F_{2D}$	$\bar{N}'$	$\bar{N}_W$	$\bar{N}_A$	$\bar{t}$ (s)
0.6	0.991	0.70	0.93	18.662	0.492	0.170	0.0035
0.7	0.953	1.11	1.41	20.318	1.752	0.566	0.0035
0.8	0.801	1.55	1.88	24.084	4.674	1.410	0.0056
0.9	0.416	1.93	2.35	29.742	9.188	2.555	0.0102
1.0	0.105	2.30	2.85	38.724	16.324	4.400	0.0130
1.1	0.008	2.58	3.44	52.875	28.500	6.375	0.0577
1.2	0.000	-	4.12	-	-	-	-
1.3	0.000	-	4.81	-	-	-	-
1.4	0.000	-	5.52	-	-	-	-

Table C.5: *Results of the embedding algorithm for* $N = 18$.

r	p	$\bar{d}^S_{2D}$	$\bar{d}^F_{2D}$	$\bar{N}'$	$\bar{N}_W$	$\bar{N}_A$	$\bar{t}$ (s)
0.6	0.987	0.80	1.10	21.033	0.782	0.251	0.0045
0.7	0.925	1.26	1.67	23.247	2.463	0.786	0.0049
0.8	0.636	1.72	2.06	28.436	6.585	1.850	0.0089
0.9	0.188	2.11	2.57	36.303	12.979	3.324	0.0173
1.0	0.012	2.37	3.19	51.083	26.000	5.083	0.0594
1.1	0.000	-	3.91	-	-	-	-
1.2	0.000	-	4.67	-	-	-	-
1.3	0.000	-	5.44	-	-	-	-
1.4	0.000	-	6.24	-	-	-	-

Table C.6: *Results of the embedding algorithm for* $N = 20$.

r	p	$\bar{d}^S_{2D}$	$\bar{d}^F_{2D}$	$\bar{N}'$	$\bar{N}_W$	$\bar{N}_A$	$\bar{t}$ (s)
0.6	0.968	0.92	1.25	23.640	1.233	0.407	0.0051
0.7	0.839	1.41	1.78	27.012	3.812	1.200	0.0067
0.8	0.425	1.88	2.25	34.200	9.553	2.647	0.0138
0.9	0.055	2.31	2.83	47.382	20.982	4.400	0.0404
1.0	0.000	-	3.57	-	-	-	-
1.1	0.000	-	4.38	-	-	-	-
1.2	0.000	-	5.22	-	-	-	-
1.3	0.000	-	6.07	-	-	-	-
1.4	0.000	-	6.97	-	-	-	-

Table C.7: *Results of the embedding algorithm for* $N = 22$.

r	p	$\bar{d}^S_{2D}$	$\bar{d}^F_{2D}$	$\bar{N}'$	$\bar{N}_W$	$\bar{N}_A$	$\bar{t}$ (s)
0.6	0.937	1.09	1.35	27.179	2.399	0.780	0.0068
0.7	0.689	1.62	1.92	33.139	7.001	2.138	0.0106
0.8	0.194	2.06	2.48	43.335	15.485	3.851	0.0255
0.9	0.011	2.50	3.18	62.000	32.182	5.818	0.0735
1.0	0.000	-	4.00	-	-	-	-
1.1	0.000	-	4.88	-	-	-	-
1.2	0.000	-	5.78	-	-	-	-
1.3	0.000	-	6.73	-	-	-	-
1.4	0.000	-	7.70	-	-	-	-

Table C.8: *Results of the embedding algorithm for* $N = 24$.

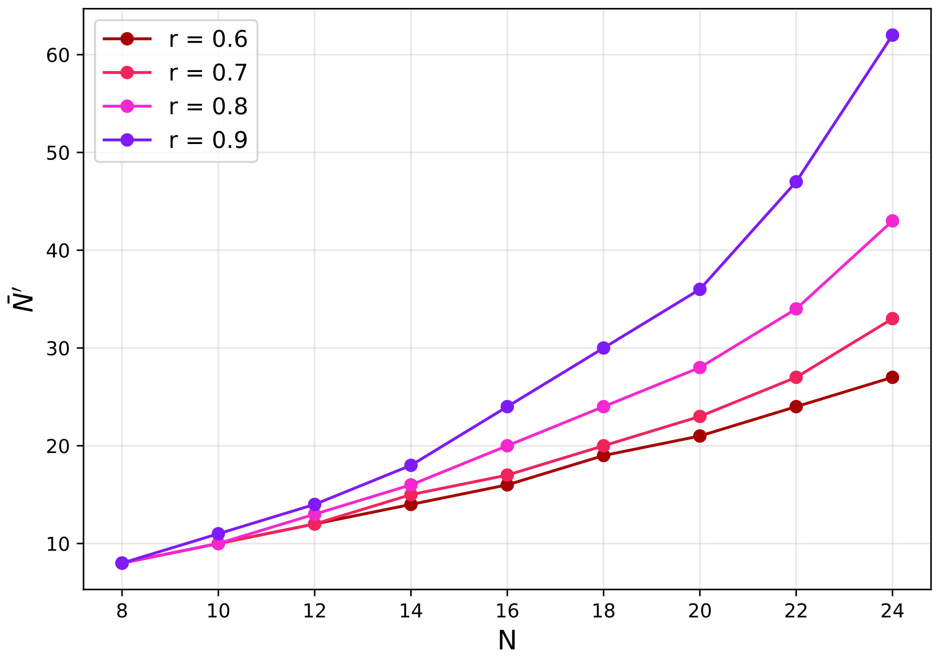


Figure C.1: *Average system size* $\bar{N}'$ *of the final configuration as a function of the original system size* N *for coverage radii* $r \in [0.6, 0.9]$. *The line connecting the markers is not a fit, just a line joining the data points, shown only to guide the eye.*

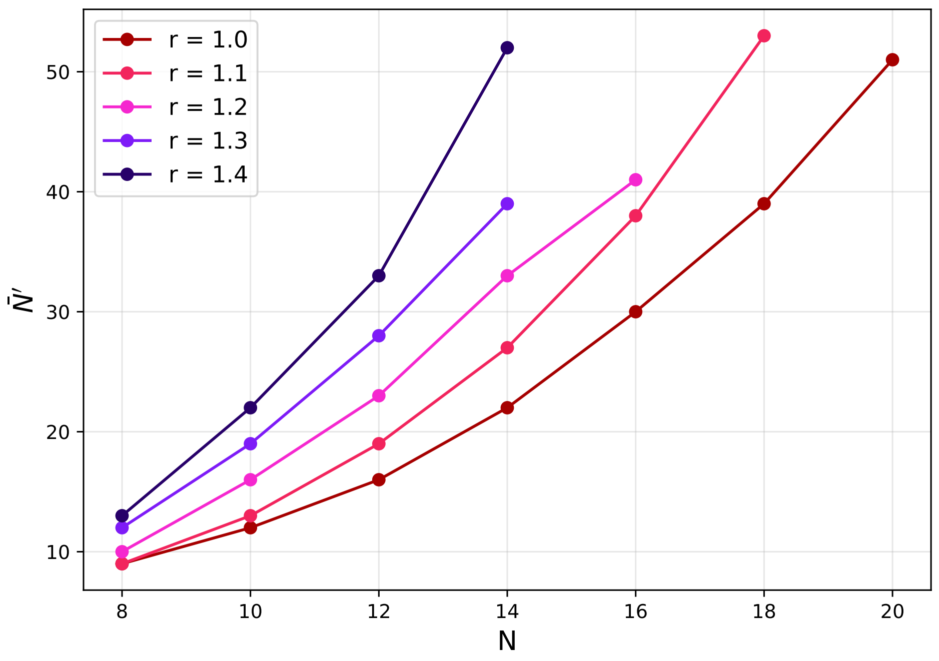


Figure C.2: *Average system size* $\bar{N}'$ *of the final configuration as a function of the original system size* N *for coverage radii* $r \in [1.0, 1.4]$. *The line connecting the markers is not a fit, just a line joining the data points, shown only to guide the eye.*

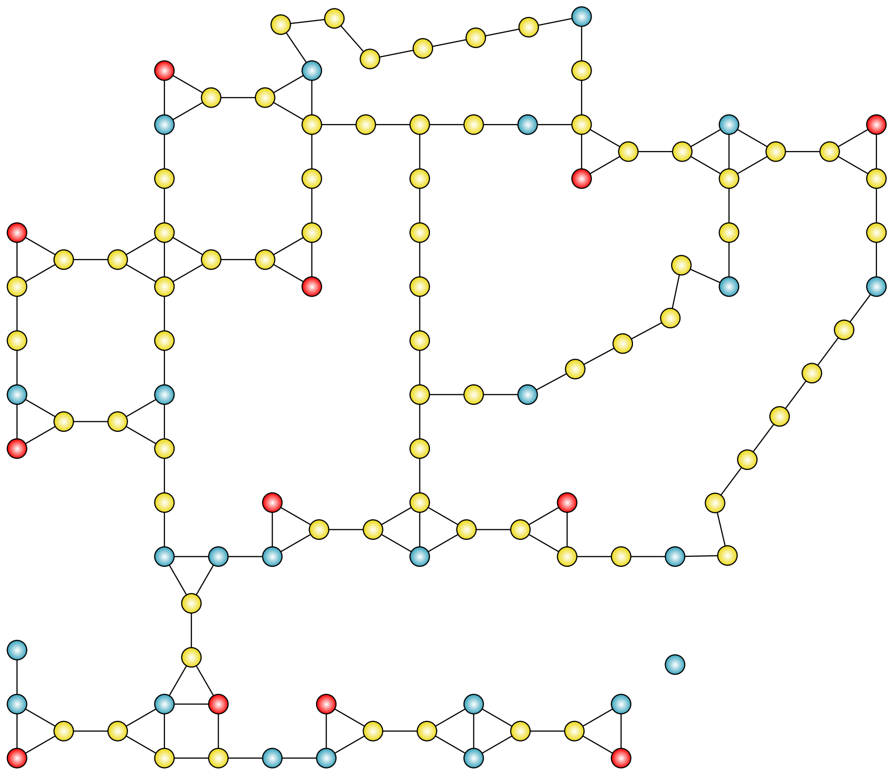


Figure C.3: *Largest embedded instance with* $N = 24$, $r = 0.9$ *and* $\bar{d} \approx 3$. *The final configuration contains* 12 *slack nodes and* 72 *wire nodes, for a total of* 108 *nodes.*

Bibliography

[1] P. Strøholm. Fermat's methods of maxima and minima and of tangents. a reconstruction. *Archive for History of Exact Sciences*, 5(1):47–69, 1968. DOI: https://doi.org/10.1007/BF00328112.

[2] J.-L. Lagrange. *Mécanique analytique.* Éditions Jacques Gabay, 1989. Facsimile reprint of the 1788 edition, ISBN 2-87647-051-9. PDF available at: https://ia.eferrit.com/ea/2be2649a90aa1e77.pdf.

[3] T. J. Ypma. Historical development of the newton–raphson method. *SIAM Review*, 37(4):531–551, 1995. DOI: https://doi.org/10.1137/1037125.

[4] C. F. Gauss. *Theoria Motus Corporum Coelestium in Sectionibus Conicis Solem Ambientium.* Cambridge University Press, 2012.

[5] L. V. Kantorovich. Mathematical methods of organizing and planning production. *Management Science*, 6(4):366–422, 1960. English translation of the 1939 Russian original. DOI: https://doi.org/10.1287/mnsc.6.4.366.

[6] G. B. Dantzig. Maximization of a linear function of variables subject to linear inequalities. In T. C. Koopmans, editor, *Activity Analysis of Production and Allocation*, number 13 in Cowles Commission Monograph, pages 339–347. Wiley, 1951. Originally appeared as a RAND report in 1947. The monograph can be found at https://cowles.yale.edu/sites/default/files/2022-09/m13-all.pdf.

[7] S. Cook. The complexity of theorem-proving procedures. *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, pages 151–158, 1971. DOI: https://doi.org/10.1145%2F800157.805047.

[8] R. M. Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations*, pages 85–103, 1972. DOI: https://doi.org/10.1007%2F978-1-4684-2001-2_9.

[9] M. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman and Company, 1979.

[10] E. Farhi, J. Goldstone, and S. Gutmann. A quantum approximate optimization algorithm. *arXiv preprint*, 2014. DOI: https://doi.org/10.48550/arXiv.1411.4028.

[11] J. Preskill. Quantum computing in the nisq era and beyond. *Quantum*, 2:79–es, 2018. DOI: https://doi.org/10.22331/q-2018-08-06-79.

[12] J. R. Finžgar et al. Quantum-informed recursive optimization algorithms. *PRX Quantum*, 5(2):020327, 2024. DOI: https://link.aps.org/doi/10.1103/PRXQuantum.5.020327.

[13] W. J. Cook. *In Pursuit of the Traveling Salesman: Mathematics at the Limits of Computation.* Princeton University Press, 2012.

[14] C.H. Papadimitriou. *Computational Complexity.* Addison-Wesley Publishing Company, Inc., 1994.

[15] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[16] B. Korte and J. Vygen. *Combinatorial Optimization: Theory and Algorithms*. Springer, 6th edition, 2018.

[17] M. Garey and D. S. Johnson. "strong" np-completeness results: Motivation, examples, and implications. *Journal of the ACM*, 25(3):499–508, 1978. DOI: https://doi.org/10.1145/322077.322090.

[18] D. Bertsimas and J. N. Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, 1997.

[19] J. Matoušek and B. Gärtner. *Understanding and Using Linear Programming*. Springer, 2007.

[20] R. Fortet. L'algebre de boole et ses applications en recherche operationnelle (in French). *Trabajos de Estadistica*, 11(2):111–118, 1960. DOI: https://doi.org/10.1007/BF03006558.

[21] P. L. Hammer and S. Rudeanu. *Boolean Methods in Operations Research and Related Areas*. Springer, 1968.

[22] F. Glover and G. Kochenberger. A unified framework for modeling and solving combinatorial optimization problems: A tutorial. In W. W. Hager et al., editor, *Multiscale Optimization Methods and Applications*, pages 101–124. Springer, 2006. DOI: https://doi.org/10.1007/0-387-29550-X_4.

[23] F. Glover and G. Kochenberger et al. The unconstrained binary quadratic programming problem: A survey. *Journal of Combinatorial Optimization*, 28:58–81, 2014. DOI: https://doi.org/10.1007/s10878-014-9734-0.

[24] Abraham P. Punnen. *The Quadratic Unconstrained Binary Optimization Problem: Theory, Algorithms, and Applications*. Springer, 2022.

[25] F. Glover, G. Kochenberger, and Y. Du. A tutorial on formulating and using qubo models. *arXiv preprint*, 2019. DOI: https://doi.org/10.48550/arXiv.1811.11538.

[26] M. Raghavachari. On connections between zero-one integer programming and concave programming under linear constraints. *Operations Research*, 17(4):680–684, 1969. DOI: https://www.jstor.org/stable/168539.

[27] M. Raghavachari. Supplement. *Operations Research*, 18(3):564–565, 1970. DOI: https://doi.org/10.1287/opre.18.3.564.

[28] B. Simon. *The Statistical Mechanics of Lattice Gases, Volume I*. Princeton University Press, 2014.

[29] S. G. Brush. History of the lenz-ising model. *Review of Modern Physics*, 39(4):883–893, 1967. DOI: https://doi.org/10.1103/RevModPhys.39.883.

[30] W. Lenz. Beiträge zum verständnis der magnetischen erscheinungen [contributions to the understanding of magnetic phenomena]. *Physikalische Zeitschrift*, 21:613–615, 1920. PDF (in German) available at https://www.physik.uni-rostock.de/storages/uni-rostock/Alle_MNF/Physik/Historisches/Kalenderblaetter_Physik/KB_2013_06_Lenz/Lenz_1920.pdf.

[31] E. Ising. Beitrag zur theorie des ferromagnetismus [contribution to the theory of ferromagnetism]. *Zeitschrift für Physik*, 31:253–258, 1925. DOI (in German): https://doi.org/10.1007/BF02980577.

[32] M. Niss. History of the lenz-ising model 1920–1950: From ferromagnetic to cooperative phenomena. *Archive for History of Exact Sciences*, 59:267–318, 2004. DOI: https://doi.org/10.1007/s00407-004-0088-3.

[33] M. Niss. History of the lenz-ising model 1950–1965: from irrelevance to relevance. *Archive for History of Exact Sciences*, 63:243–287, 2008. DOI: https://doi.org/10.1007/s00407-008-0039-5.

[34] M. Niss. History of the lenz-ising model 1965–1971: the role of a simple model in understanding critical phenomena. *Archive for History of Exact Sciences*, 65:625–658, 2011. DOI: https://doi.org/10.1007/s00407-011-0086-1.

[35] E. Ising et al. The fate of ernst ising and the fate of his model. *Journal of Physical Studies*, 21(3), 2017. Article 3002, 19 pp. DOI: https://doi.org/10.30970/jps.21.3002.

[36] A. Lucas. Ising formulations of many np problems. *Frontiers in Physics*, 2(5), 2014. DOI: https://doi.org/10.3389/fphy.2014.00005.

[37] J.J. Balmer. Notiz über die spektrallinien des wasserstoffs. *Annalen der Physik und Chemie*, 25:548–560, 1885. DOI: https://doi.org/10.3931/e-rara-134837.

[38] J. J. Balmer. Note on the spectral lines of hydrogen. In H. A. Boorse and L. Motz, editors, *The World of the Atom, Vol. 1*, pages 33–36. Basic Books, 1966. English translation of Balmer's 1885 paper. PDF available at: https://www.ub.edu/hcub/hfq/sites/default/files/Balmer_1885.pdf.

[39] T. F. Gallagher. *Rydberg Atoms*. Cambridge University Press, 2005.

[40] J. R. Rydberg. Researches sur la constitution des spectres d'émission des éléments chimiques [investigations of the composition of the emission spectra of chemical elements]. *Kungliga Svenska Vetenskaps-Akademiens Handlingar [Proceedings of the Royal Swedish Academy of Science], 2nd edition*, 23(11):1–155, 1889. Available (in French) at https://babel.hathitrust.org/cgi/pt?id=mdp.39015039478303.

[41] M. Saffman, T. G. Walker, and K. Mølmer. Quantum information with rydberg atoms. *Reviews of Modern Physics*, 82(3):2313–2363, 2010. DOI: https://link.aps.org/doi/10.1103/RevModPhys.82.2313.

[42] M. Morgado and S. Whitlock. Quantum simulation and computing with rydberg-interacting qubits. *AVS Quantum Science*, 3(2):023501, 2021. DOI: https://doi.org/10.1116/5.0036562.

[43] J. Stark. Beobachtungen über den effekt des elektrischen feldes auf spektrallinien. i. quereffekt. *Annalen der Physik*, 43(7):965–982, 1913. DOI (in German): https://doi.org/10.1002/andp.19143480702.

[44] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2010.

[45] L. Henriet et al. Quantum computing with neutral atoms. *Quantum*, 4:327, 2020. DOI: https://doi.org/10.22331/q-2020-09-21-327.

[46] M. Grotti, S. Marzella, G. Bettonte, D. Ottaviani, and E. Ercolessi. Practical use cases of neutral atoms quantum computers. *arXiv preprint*, 2025. DOI: https://doi.org/10.48550/arXiv.2510.18732.

[47] L. K. Grover. A fast quantum mechanical algorithm for database search. *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing - STOC '96*, pages 212–219, 1996. DOI: https://doi.org/10.1145/237814.237866.

[48] M. Lubasch et al. Variational quantum algorithms for nonlinear problems. *Physical Review A*, 101(1):010301, 2020. DOI: https://link.aps.org/doi/10.1103/PhysRevA.101.010301.

[49] A. Parra-Rodriguez et al. Digital-analog quantum computation. *Physical Review A*, 101(2):022305, 2020. DOI: https://link.aps.org/doi/10.1103/PhysRevA.101.022305.

[50] X. Wu et al. A concise review of rydberg atom based quantum computation and quantum simulation. *Chinese Physics B*, 30(2):020305, 2021. DOI: https://doi.org/10.1088/1674-1056/abd76f.

[51] J. Wurtz et al. Aquila: Quera's 256-qubit neutral-atom quantum computer. *arXiv preprint*, 2023. DOI: https://doi.org/10.48550/arXiv.2306.11727.

[52] G. S. Agarwal. Vacuum-field rabi oscillations of atoms in a cavity. *Journal of the Optical Society of America B*, 2(3):480–485, 1985. DOI: https://doi.org/10.1364/JOSAB.2.000480.

[53] T. Förster. Zwischenmolekulare energiewanderung und fluoreszenz. *Annalen der Physik*, 437(1-2):55–75, 1948. DOI (in German): https://doi.org/10.1002%2Fandp.19484370105.

[54] T. Förster. Energy migration and fluorescence. *Journal of Biomedical Optics*, 17(1):011002, 2012. English translation of Förster's original work. DOI: https://doi.org/10.1117/1.JBO.17.1.011002.

[55] M. D. Lukin et al. Dipole blockade and quantum information processing in mesoscopic atomic ensembles. *Physical Review Letters*, 87:037901, 2001. DOI: https://doi.org/10.1103/PhysRevLett.87.037901.

[56] A. Browaeys and T. Lahaye. Many-body physics with individually-controlled rydberg atoms. *Nature Physics*, 16:132–142, 2020. DOI: https://doi.org/10.1038/s41567-019-0733-z.

[57] D. A. Steck. Rubidium 87 d line data, 2003. PDF available at: https://steck.us/alkalidata/rubidium87numbers.1.6.pdf.

[58] E. L. Raab, M. Prentiss, A. Cable, S. Chu, and D. E. Pritchard. Trapping of neutral sodium atoms with radiation pressure. *Physical Review Letters*, 59(23):2631–2634, 1987. DOI: https://doi.org/10.1103/PhysRevLett.59.2631.

[59] J. Dalibard and C. Cohen-Tannoudji. Laser cooling below the doppler limit by polarization gradients: simple theoretical models. *Journal of the Optical Society of America B*, 6(11):2023–2045, 1989. DOI: https://doi.org/10.1364/JOSAB.6.002023.

[60] K. Krzysztof et al. Magneto-optical trap: Fundamentals and realization. *Computational Methods in Science and Technology*, Special Issue(2):115–129, 2010. DOI: http://dx.doi.org/10.12921/cmst.2010.SI.02.115-129.

[61] A. Ashkin. Acceleration and trapping of particles by radiation pressure. *Physical Review Letters*, 24:156–159, 1970. DOI: https://link.aps.org/doi/10.1103/PhysRevLett.24.156.

[62] A. Ashkin, J. M. Dziedzic, J. E. Bjorkholm, and S. Chu. Observation of a single-beam gradient force optical trap for dielectric particles. *Optics Letters*, 11(5):288–290, 1986. DOI: https://doi.org/10.1364/OL.11.000288.

[63] D. Barredo et al. An atom-by-atom assembler of defect-free arbitrary two-dimensional atomic arrays. *Science*, 354(6315):1021–1023, 2016. DOI: https://doi.org/10.1126/science.aah3778.

[64] X. F. Shi. Single-site rydberg addressing in 3d atomic arrays for quantum computing with neutral atoms. *Journal of Physics B: Atomic, Molecular and Optical Physics*, 53(5):054002, 2020. DOI: https://doi.org/10.1088/1361-6455/ab5f79.

[65] W. S. Bakr. A quantum gas microscope for detecting single atoms in a hubbard-regime optical lattice. *Nature*, 462:74–77, 2009. DOI: https://doi.org/10.1038/nature08482.

[66] M. J. Gibbons et al. Nondestructive fluorescent state detection of single neutral atom qubits. *Physical Review Letters*, 106(13):133002, 2011. DOI: https://link.aps.org/doi/10.1103/PhysRevLett.106.133002.

[67] M. Martinez-Dorantes et al. Fast nondestructive parallel readout of neutral atom registers in optical potentials. *Physical Review Letters*, 119(18):180503, 2017. DOI: https://link.aps.org/doi/10.1103/PhysRevLett.119.180503.

[68] D. Barredo et al. Synthetic three-dimensional atomic structures assembled atom by atom. *Nature*, 561:79–82, 2018. DOI: https://doi.org/10.1038/s41586-018-0450-2.

[69] H. Weimer, M. Müller, I. Lesanovsky, P. Zoller, and H. P. Büchler. A rydberg quantum simulator. *Nature Physics*, 6:382–388, 2010. DOI: https://doi.org/10.1038/nphys1614.

[70] H. Bernien et al. Probing many-body dynamics on a 51-atom quantum simulator. *Nature*, 551:579–584, 2017. DOI: https://doi.org/10.1038/nature24622.

[71] D. P. Fahey and M. W. Noel. Excitation of rydberg states in rubidium with near infrared diode lasers. *Optics Express*, 19(18):17002–17012, 2011. DOI: https://doi.org/10.1364/OE.19.017002.

[72] D. Tong et al. Local blockade of rydberg excitation in an ultracold gas. *Physical Review Letters*, 93(6):063001, 2004. DOI: https://doi.org/10.1103/PhysRevLett.93.063001.

[73] J. Wang, J. Bai, J. He, and J. Wang. Single-photon cesium rydberg excitation spectroscopy using 318.6-nm uv laser and room-temperature vapor cell. *Optics Express*, 25(19):22510–22518, 2017. DOI: https://doi.org/10.1364/OE.25.022510.

[74] S. A. Lee, J. Helmcke, J. L. Hall, and B. P. Stoicheff. Doppler-free two-photon transitions to rydberg levels: convenient, useful, and precise reference wavelengths for dye lasers. *Optics Letters*, 3(4):141–143, 1978. DOI: https://doi.org/10.1364/OL.3.000141.

[75] Y. Miroshnychenko et al. Coherent excitation of a single atom to a rydberg state. *Physical Review A*, 82(1):013405, 2010. DOI: https://link.aps.org/doi/10.1103/PhysRevA.82.013405.

[76] A. Das and B. K. Chakrabarti. Quantum annealing and analog quantum computation. *Reviews of Modern Physics*, 80(3):1061–1081, 2008. DOI: https://link.aps.org/doi/10.1103/RevModPhys.80.1061.

[77] M. Born and V. Fock. Beweis des adiabatensatzes [proof of the adiabatic theorem]. *Zeitschrift für Physik*, 51(3):165–180, 1928. DOI (in German): https://doi.org/10.1007/BF01343193.

[78] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser. Quantum computation by adiabatic evolution. *arXiv preprint*, 2000. DOI: https://doi.org/10.48550/arXiv.quant-ph/0001106.

[79] D. Aharonov et al. Adiabatic quantum computation is equivalent to standard quantum computation. *SIAM Journal of Computing*, 37(1):166–194, 2007. DOI: https://doi.org/10.1137/S0097539705447323.

[80] J. D. Biamonte and P. J. Love. Realizable hamiltonians for universal adiabatic quantum computers. *Physical Review A*, 78(1):012352, 2008. DOI: https://link.aps.org/doi/10.1103/PhysRevA.78.012352.

[81] T. Albash and D. A. Lidar. Adiabatic quantum computation. *Reviews of Modern Physics*, 90(1):015002, 2018. DOI: https://link.aps.org/doi/10.1103/RevModPhys.90.015002.

[82] L. Phuttitarn et al. Enhanced measurement of neutral atom qubits with machine learning. *Physical Review Applied*, 22(2):024011, 2024. DOI: https://link.aps.org/doi/10.1103/PhysRevApplied.22.024011.

[83] R. M. Kent et al. Efficient measurement of neutral-atom qubits with matched filters. *arXiv preprint*, 2025. DOI: https://doi.org/10.48550/arXiv.2504.08170.

[84] P. Aliferis and B. M. Terhal. Fault-tolerant quantum computation for local leakage faults. *Quantum Information & Computation*, 7(1):139–156, 2007. DOI: https://doi.org/10.26421/QIC7.1-2-9.

[85] F. Motzoi, J. M. Gambetta, P. Rebentrost, and F. K. Wilhelm. Simple pulses for elimination of leakage in weakly nonlinear qubits. *Physical Review Letters*, 103(11):110501, 2009. DOI: https://link.aps.org/doi/10.1103/PhysRevLett.103.110501.

[86] M. Sarovar et al. Detecting crosstalk errors in quantum information processors. *Quantum*, 4:321, 2020. DOI: https://doi.org/10.22331/q-2020-09-11-321.

[87] E. Magesan and J. M. Gambetta. Effective hamiltonian models of the cross-resonance gate. *Physical Review A*, 101(5):052308, 2020. DOI: https://link.aps.org/doi/10.1103/PhysRevA.101.052308.

[88] D. Bluvstein et al. A quantum processor based on coherent transport of entangled atom arrays. *Nature*, 604:451–456, 2022. DOI: https://doi.org/10.1038/s41586-022-04592-6.

[89] I. Cong et al. Hardware-efficient, fault-tolerant quantum computation with rydberg atoms. *Physical Review X*, 12:021049, 2022. DOI: https://doi.org/10.1103/PhysRevX.12.021049.

[90] S. K. Barik et al. Quantum technologies with rydberg atoms. *Frontiers in Quantum Science and Technology*, 3:1426216, 2024. DOI: https://doi.org/10.3389/frqst.2024.1426216.

[91] M. Saffman. Quantum computing with atomic qubit arrays: confronting the cost of connectivity. *arXiv preprint*, 2025. DOI: https://doi.org/10.48550/arXiv.2505.11218.

[92] S. Sunami et al. Scalable networking of neutral-atom qubits: Nanofiber-based approach for multiprocessor fault-tolerant quantum computer. *PRX Quantum*, 6:010101, 2025. DOI: https://doi.org/10.1103/PRXQuantum.6.010101.

[93] Pasqal. *The Power of Neutral Atom Quantum Processors*, 2025. Company white paper detailing QPU specifications and future roadmap. PDF available at: https://www.pasqal.com/wp-content/uploads/2025/10/2025-Pasqal-Quantum-Computing-Processor-Brochure.pdf.

[94] M. Vandelli, A. Lignarolo, C. Cavazzoni, and D. Dragoni. Evaluating the practicality of quantum optimization algorithms for prototypical industrial applications. *Quantum Information Processing*, 23:344, 2024. DOI: https://doi.org/10.1007/s11128-024-04560-1.

[95] I. B. Vapnyarskii. Lagrange multipliers, 2001 [1994]. Encyclopedia of Mathematics, EMS Press. https://encyclopediaofmath.org/wiki/Lagrange_multipliers.

[96] G. Colucci, S. van der Linde, and F. Phillipson. Power network optimization: a quantum approach. *arXiv preprint*, 2022. DOI: https://doi.org/10.48550/arXiv.2212.01625.

[97] F. L. Gewers eet al. Principal component analysis: A natural approach to data exploration. *ACM Computing Surveys (CSUR)*, 54(4):1–34, 2021. DOI: https://doi.org/10.1145/3447755.

[98] R. A. Horn and C. R. Johnson. *Matrix Analysis, 2nd edition*. Cambridge University Press, 2013.

[99] C. Eckart and G. Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936. DOI: https://doi.org/10.1007/BF02288367.

[100] L. Mirsky. Symmetric gauge functions and unitarily invariant norms. *The Quarterly Journal of Mathematics*, 11(1):50–59, 1960. DOI: https://doi.org/10.1093/qmath/11.1.50.

[101] A. Byun, M. Kim, and J. Ahn. Finding the maximum independent sets of platonic graphs using rydberg atoms. *PRX Quantum*, 3(3):03035, 2022. DOI: https://doi.org/10.1103/PRXQuantum.3.030305.

[102] X. Qiu, P. Zoller, and X. Li. Programmable quantum annealing architectures with ising quantum wires. *PRX Quantum*, 1(2):020311, 2020. DOI: https://link.aps.org/doi/10.1103/PRXQuantum.1.020311.

[103] M. Kim et al. Rydberg quantum wires for maximum independent set problems with nonplanar and high-degree graphs. *Nature Physics*, 18(7):755–759, 2022. DOI: https://doi.org/10.1038/s41567-022-01629-5.

[104] K.-N. Schymik et al. Single atoms with 6000-second trapping lifetimes in optical-tweezer arrays at cryogenic temperatures. *Physical Review Applied*, 16(3):034013, 2021. DOI: https://link.aps.org/doi/10.1103/PhysRevApplied.16.034013.

[105] I. S. Madjarov et al. High-fidelity entanglement and detection of alkaline-earth rydberg atoms. *Nature Physics*, 16(18):857–861, 2020. DOI: https://doi.org/10.1038/s41567-020-0903-z.

[106] A. G. de Oliveira et al. Demonstration of weighted-graph optimization on a rydberg-atom array using local light shifts. *PRX Quantum*, 6(1):010301, 2025. DOI: https://link.aps.org/doi/10.1103/PRXQuantum.6.010301.

[107] J. Kombe et al. A quantum wire approach to weighted combinatorial graph optimisation problems. *arXiv preprint*, 2025. DOI: https://doi.org/10.48550/arXiv.2503.17115.

[108] Y. Yang, A. Forbes, and L. Cao. A review of liquid crystal spatial light modulators: devices and applications. *Opto-Electronic Science*, 2(8):230026, 2023. DOI: https://doi.org/10.29026/oes.2023.230026.

[109] B. N. Clark, C. J. Colbourn, and D. S. Johnson. Unit disk graphs. *Discrete Mathematics*, 86(1):165–177, 1990. DOI: https://doi.org/10.1016/0012-365X(90)90358-O.

[110] R. L. Cook. Stochastic sampling in computer graphics. *ACM Transactions on Graphics*, 5(1):51–72, 1986. DOI: https://doi.org/10.1145/7529.8927.

[111] R. Bridson. Fast poisson disk sampling in arbitrary dimensions. In *ACM SIGGRAPH 2007 Sketches*, pages 22–es, 2007. DOI: https://doi.org/10.1145/1278780.1278807.

[112] S. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982. DOI: https://doi.org/10.1109/TIT.1982.1056489.

[113] B. Duhaime. Lloyd's algorithm. https://github.com/duhaime/lloyd, 2019.

[114] J. H. Halton. Algorithm 247: Radical-inverse quasi-random point sequence. *Communications of the ACM*, 7(12):701–702, 1964. DOI: https://doi.org/10.1145/355588.365104.

[115] I.M Sobol'. On the distribution of points in a cube and the approximate evaluation of integrals. *USSR Computational Mathematics and Mathematical Physics*, 7(4):86–112, 1967. DOI: https://doi.org/10.1016/0041-5553(67)90144-9.

[116] L. Kuipers and H. Niederreiter. *Uniform Distribution of Sequences*. John Wiley & Sons Inc., 1974.

[117] Plotly Technologies Inc. Plotly: Python graphing library. https://plotly.com/python/, 2024.

[118] R. Vidali. Embedding algorithm. https://github.com/UniboRick/embedding-algorithm, 2025.

[119] K. Bharti et al. Noisy intermediate-scale quantum algorithms. *Reviews of Modern Physics*, 94(1):015004, 2022. DOI: https://link.aps.org/doi/10.1103/RevModPhys.94.015004.

[120] J.R. McClean et al. The theory of variational hybrid quantum-classical algorithms. *New Journal of Physics*, 18(2):023023, 2016. DOI: https://doi.org/10.1088/1367-2630/18/2/023023.

[121] K. Blekos et al. A review on quantum approximate optimization algorithm and its variants. *Physics Reports*, 1068:1–66, 2024. DOI: https://doi.org/10.1016/j.physrep.2024.03.002.

[122] M. Cerezo et al. Variational quantum algorithms. *Nature Reviews Physics*, 3:625–644, 2021. DOI: https://doi.org/10.1038/s42254-021-00348-9.

[123] F. M. Fernández. On the rayleigh-ritz variational method. *arXiv preprint*, 2022. DOI: https://doi.org/10.48550/arXiv.2206.05122.

[124] E. Farhi et al. A quantum adiabatic evolution algorithm applied to random instances of an np-complete problem. *Science*, 92(5516):472–475, 2001. DOI: https://doi.org/10.1126/science.1057726.

[125] H. F. Trotter. On the product of semi-groups of operators. *Proceedings of the American Mathematical Society*, 10(4):545–551, 1959. DOI: https://doi.org/10.2307/2033649.

[126] M. Suzuki. Generalized trotter's formula and systematic approximants of exponential operators and their applications to many-body problems. *Communications in Mathematical Physics*, 51(2):183–190, 1976. DOI: https://doi.org/10.1007/BF01609348.

[127] S. Gutmann E. Farhi, D. Gamarnik. The quantum approximate optimization algorithm needs to see the whole graph: A typical case. *arXiv preprint*, 2020. DOI: https://doi.org/10.48550/arXiv.2004.09002.

[128] S. Gutmann E. Farhi, D. Gamarnik. The quantum approximate optimization algorithm needs to see the whole graph: Worst case examples. *arXiv preprint*, 2020. DOI: https://doi.org/10.48550/arXiv.2005.08747.

[129] S. Bravyi et al. Obstacles to variational quantum optimization from symmetry protection. *Physical Review Letters*, 125(26):260505, 2020. DOI: https://link.aps.org/doi/10.1103/PhysRevLett.125.260505.

[130] S. Bravyi et al. Hybrid quantum-classical algorithms for approximate graph coloring. *Quantum*, 6:678–es, 2022. DOI: https://doi.org/10.22331/q-2022-03-30-678.

[131] J. R. Finžgar. Qiro. https://github.com/jernejrudifinzgar/qiro, 2023.

[132] J.-G. Liu et al. Computing solution space properties of combinatorial optimization problems via generic tensor networks. *SIAM Journal on Scientific Computing*, 45(3):A1239–A1270, 2023. DOI: https://doi.org/10.1137/22M1501787.

[133] S. Ebadi et al. Quantum optimization of maximum independent set using rydberg atom arrays. *Science*, 376(6598):1209–1215, 2022. DOI: https://www.science.org/doi/abs/10.1126/science.abo6587.

[134] M. M. Halldórsson and J. Radhakrishnan. Greed is good: Approximating independent sets in sparse and bounded-degree graphs. *Algorithmica*, 18(1):145–183, 1997. DOI: https://doi.org/10.1007/BF02523693.

[135] C. Bron and J. Kerbosch. Algorithm 457: finding all cliques of an undirected graph. *Communications of the ACM*, 16(9):575–577, 1973. DOI: https://doi.org/10.1145/362342.362367.

[136] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, Third Edition*. MIT Press, 2009.

[137] G. Parisi. Order parameter for spin-glasses. *Physical Review Letters*, 50(24):1946–1948, Jun 1983. DOI: https://doi.org/10.1103/PhysRevLett.50.1946.

[138] E. Schrödinger. Probability relations between separated systems. *Mathematical Proceedings of the Cambridge Philosophical Society*, 32(3):446–452, 1936. DOI: https://doi.org/10.1017%2FS0305004100019137.

[139] L. P. Hughston, R. Jozsa, and W. K. Wootters. A complete classification of quantum ensembles having a given density matrix. *Physics Letters A*, 183(1):14–18, 1993. DOI: https://doi.org/10.1016%2F0375-9601%2893%2990880-9.

[140] W. K. Wootters and W. H. Zurek. A single quantum cannot be cloned. *Nature*, 299:802–803, 1982. DOI: https://doi.org/10.1038/299802a0.

[141] S. Vadhan. Computational complexity. *Encyclopedia of Cryptography and Security*, page 235–240, 2011. DOI: https://doi.org/10.1007/978-1-4419-5906-5_442.

[142] A. M. Turing. On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London Mathematical Society*, 42(2):230–265, 1936. DOI: https://doi.org/10.1112/plms/s2-42.1.230.

[143] S. Arora and B. Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.

[144] J. C. Martin. *Introduction to Languages and The Theory of Computation*. McGraw-Hill Science/Engineering/Math, 2010.

[145] A. Church. A set of postulates for the foundation of logic. *Annals of Mathematics*, 33(2):346–366, 1932. DOI: https://doi.org/10.2307%2F1968337.

[146] A. M. Turing. Computability and λ-definability. *The Journal of Symbolic Logic*, 2(4):153–163, 1937. DOI: https://doi.org/10.2307%2F2268280.

[147] A. Church. An unsolvable problem of elementary number theory. *American Journal of Mathematics*, 58(2):345–363, 1936. DOI: https://doi.org/10.2307%2F2371045.

[148] K. Gödel. On undecidable propositions of formal mathematical systems. In *Kurt Gödel: Collected Works, Volume I: Publications 1929-1936*, pages 346–371. Oxford University Press, 1986. DOI: https://doi.org/10.1093/oso/9780195147209.001.0001.

[149] S. C. Kleene. General recursive functions of natural numbers. *Mathematische Annalen*, 112:727–741, 1936. DOI: https://doi.org/10.1007/BF01565439.

[150] A. M. Ben-Amram. The church-turing thesis and its look-alikes. *SIGACT News*, 36(3):113–116, 2005. DOI: https://doi.org/10.1145%2F1086649.1086651.

[151] M. Sipser. *Introduction to the Theory of Computation*. Cengage Learning, 2013.

[152] R. de Wolf. Quantum computing: Lecture notes, 2019. https://arxiv.org/abs/1907.09415.

[153] J. Van Leeuwen. *Handbook of Theoretical Computer Science, Vol. A: Algorithms and Complexity*. MIT Press, 1990.

[154] L. A. Levin. Universal sequential search problems. *Problems of Information Transmission*, 9(3):115–116, 1973.

[155] B. A. Trakhtenbrot. A survey of russian approaches to perebor (brute-force searches) algorithms. *Annals of the History of Computing*, 6(4):384–400, 1984. Includes an English translation of Levin's 1973 paper *Universal Sequential Search Problems* in the appendix. DOI: https://doi.org/10.1109%2FMAHC.1984.10036.

[156] L. Fortnow. Fifty years of p versus np and the possibility of the impossible. *Communications of the ACM*, 65(1):76–85, 2021. DOI: https://doi.org/10.1145/3460351.

[157] F. Vega. Sat in polynomial time: A proof of p = np. *engrXiv preprint*, 2024. DOI: https://doi.org/10.31224/4189.

[158] T. Lin. The separation of np and pspace. *arXiv preprint*, 2021. DOI: https://doi.org/10.48550/arXiv.2106.11886.

[159] J. Gill. Computational complexity of probabilistic turing machines. *SIAM Journal on Computing*, 6(4):675–695, 1977. DOI: https://doi.org/10.1137/0206049.

[160] S. Chawla. Randomized algorithms, lecture 2: Complexity classes, 2004. Lecture notes for 15-859(M), September 15, 2004. Scribe: Andrew Gilpin. PDF available at https://www.cs.cmu.edu/afs/cs/academic/class/15859-f04/www/scribes/lec2.pdf.

[161] R. O'Donnell. Quantum computation, lecture 23: Introduction to quantum complexity theory, 2015. Lecture notes, Fall 2015. Scribe: Will Griffin. PDF available at https://www.cs.cmu.edu/~odonnell/quantum15/lecture23.pdf.

[162] C. H. Bennett, E. Bernstein, G. Brassard, and U. Vazirani. Strengths and weaknesses of quantum computing. *SIAM Journal on Computing*, 26(5):1510–1523, 1997. DOI: https://doi.org/10.1137/S0097539796300933.

[163] S. Gharibian and J. Kamminga. Bqp, meet np: Search-to-decision reductions and approximate counting. *51st International Colloquium on Automata, Languages, and Programming*, 297:70:1–70:19, 2024. DOI: https://doi.org/10.4230/LIPIcs.ICALP.2024.70.

[164] G. J. Woeginger. Exact algorithms for np-hard problems: A survey. In Michael Jünger, Gerhard Reinelt, and Giovanni Rinaldi, editors, *Combinatorial Optimization - Eureka, You Shrink!*, volume 2570 of *Lecture Notes in Computer Science*, pages 185–207. Springer, 2003. DOI: https://doi.org/10.1007/3-540-36478-1_17.

[165] W. Li et al. Parameterized algorithms of fundamental np-hard problems: a survey. *Human-centric Computing and Information Sciences*, 10(29), 2020. DOI: https://doi.org/10.1186/s13673-020-00226-w.

[166] D. S. Hochbaum. *Approximation Algorithms for NP-Hard Problems.* Course Technology, 1996.

[167] T. Dokeroglu, D. Canturk, and T. Kucukyilmaz. A survey on pioneering metaheuristic algorithms between 2019 and 2024. *arXiv preprint*, 2024. DOI: https://doi.org/10.48550/arXiv.2501.14769.