Pontifícia Universidade Católica
do Rio de Janeiro

Daniel Ribas Tandeitnik

Evolving Quantum Error Correction Codes

Dissertação de Mestrado

Dissertation presented to the Programa de Pós-graduação em Física of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Ciências - Física.

Advisor: Prof. Thiago Barbosa dos Santos Guerreiro

Rio de Janeiro
May 2022

Pontifícia Universidade Católica
do Rio de Janeiro

Daniel Ribas Tandeitnik

Evolving Quantum Error Correction Codes

Dissertation presented to the Programa de Pós-graduação em Física of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Ciências - Física. Approved by the Examination Committee:

Prof. Thiago Barbosa dos Santos Guerreiro
Advisor
Departamento de Física – PUC-Rio

Prof. Guilherme Penello Temporão
CETUC – PUC-Rio

Prof. Stuart Kauffman
UPENN

Prof. Ernesto Fagundes Galvão
UFF

Rio de Janeiro, May 11th, 2022

Daniel Ribas Tandeitnik

The author graduated in Physics from Pontifícia Universidade Católica do Rio de Janeiro in 2019.

Bibliographic data

Ribas Tandeitnik, Daniel

Evolving Quantum Error Correction Codes / Daniel Ribas Tandeitnik; advisor: Thiago Barbosa dos Santos Guerreiro. – 2022.

92 f: il. color. ; 30 cm

Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Física, 2022.

Inclui bibliografia

1. Física - Teses. 2. algoritmo genético. 3. correção de erros quântico. 4. códigos stabilizer.
I. Barbosa dos Santos Guerreiro, Thiago. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Física. III. Título.

CDD: 530

To all the people who supported me and made it possible, directly and indirectly, for me to take the road less traveled.

Let there be quantum circuits!

Acknowledgments

We thank Bruno Suassuna and Igor Brandão for useful discussions. This study was financed in part by Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), and by the Fundação de Amparo à Pesquisa do Estado do Rio de Janeiro (FAPERJ). This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001. I would like to thank the support received by CNPq Scholarship No. 132606/2020-8 and FAPERJ Scholarship No. 2021.01394.9.

Abstract

Ribas Tandeitnik, Daniel; Barbosa dos Santos Guerreiro, Thiago (Advisor). **Evolving Quantum Error Correction Codes**. Rio de Janeiro, 2022. 92p. Dissertação de Mestrado – Departamento de Física, Pontifícia Universidade Católica do Rio de Janeiro.

Computational methods become essential in the face of complex problems where human intuition and traditional methods fail. Recent works present artificial neural networks capable of efficiently performing tasks intractable by conventional algorithms using machine learning, rendering it one of the most popular methods. Concomitantly, genetic algorithms, inspired by the biological processes of natural selection and mutation, have been used as a metaheuristic method to find solutions to optimization problems. We then raise the question of whether genetic algorithms have the potential to solve problems in the context of quantum computing, where human intuition decreases as physical systems grow. Specifically, we focus on the evolution of quantum error-correcting codes within the stabilizer code formalism. By specifying an appropriate fitness function, we show that we can evolve celebrated codes, such as the Perfect and Shor's code with respectively 5 and 9 qubits, in addition to new unanticipated examples. Additionally, we compared it with a brute force random search and verified an increasing superiority of the genetic algorithm as the total number of qubits increases. Given the results, we foresee that genetic algorithms can become valuable tools to perform complex applications in quantum systems and produce tailored circuits that satisfy restrictions imposed by hardware.

Keywords

genetic algorithm; quantum error correction; stabilizer codes;.

Resumo

Ribas Tandeitnik, Daniel; Barbosa dos Santos Guerreiro, Thiago. **Evoluindo Códigos de Correção de Erros Quânticos**. Rio de Janeiro, 2022. 92p. Dissertação de Mestrado – Departamento de Física, Pontifícia Universidade Católica do Rio de Janeiro.

Métodos computacionais se tornam essenciais diante de problemas complexos onde a intuição humana e métodos tradicionais falham. Trabalhos recentes apresentam redes neurais artificiais capazes de realizar eficientemente tarefas intratáveis por algoritmos convencionais com o emprego de aprendizado de máquina, tornando-se assim um dos métodos mais populares. Concomitantemente, algoritmos genéticos, inspirados pelos processos biológicos de seleção natural e mutação, têm sido utilizados como método metaheurístico para encontrar soluções de problemas de otimização. Levantamos então a questão se algoritmos genéticos possuem potencial para resolver problemas no contexto da computação quântica, onde a intuição humana decresce à medida que os sistemas físicos crescem. Especificamente, nos concentramos na evolução de códigos de correção de erros quânticos dentro do formalismo de códigos *stabilizer*. Ao especificar uma função de *fitness* apropriada, mostramos que somos capazes de evoluir códigos celebrados, como o código do Shor e o perfeito de 9 e 5 qubits respectivamente, além de novos exemplos não antecipados. Adicionalmente, comparamos com o método força bruta de busca aleatória e verificamos uma crescente superioridade do algoritmo genético conforme aumenta-se o número total de qubits. Diante dos resultados, imaginamos que algoritmos genéticos possam se tornar ferramentas valiosas para desempenhar aplicações complexas em sistemas quânticos e produzir circuitos sob medida que satisfaçam restrições impostas por *hardware*.

Palavras-chave

algoritmo genético; correção de erros quântico; códigos stabilizer;.

Table of contents

List of figures

List of tables

List of Abreviations

EC – encoding circuit

ES – evolution strategies

GA – genetic algorithm

NISQ – noisy intermediate-scale quantum

QECC – quantum error correction code

RS – random search

Astronauta libertado
Minha vida me ultrapassa
Em qualquer rota que eu faça
Dei um grito no escuro
Sou parceiro do futuro
Na reluzente galáxia

Eu quase posso falar
A minha vida é que grita
Emprenha se reproduz
Na velocidade da luz
A cor do sol me compõe
O mar azul me dissolve
A equação me propõe
Computador me resolve

Astronauta libertado
Minha vida me ultrapassa
Em qualquer rota que eu faça
Dei um grito no escuro
Sou parceiro do futuro
Na reluzente galáxia

Antonio Jose Santana Martins / Rita Lee Jones De Carvalho,
Fragmento da letra de Dois mil e um © Warner Chappell Music, Inc.

1
Introduction

Why evolve quantum circuits

Natural selection is a unifying idea in biology [1]. Through reproduction, mutation and competition, systems can navigate the complexity landscape [2] from small-sized molecules and molecule sets [3] to complex molecular machines [4] to living organisms [5]. Just like natural selection enables the rise of complex designs such as eyes and brains in the universe [6], artificial evolution can be employed in the laboratory to produce molecules with desired optical response properties [7]. In view of that, an interesting engineering question is whether one can exploit evolution as a design tool for complex technology where human intuition and the standard deductive method might encounter difficulties or perhaps even fail.

A natural field in which human intuition often encounters difficulties is quantum information science. Devising innovative means of producing complex quantum states and algorithms [8] outside the scope of known quantum information primitives [9] is a challenging task [10], especially under the limitations imposed by present-day quantum hardware [11]. This motivates the main question of this work: *Can artificial selection be employed as an effective tool to design useful quantum circuits?*

Computer-assisted searches for new physical phenomena [12] and laws of nature [13–15] comprise a very timely research topic, with notable examples including the search for new quantum optics experiments [16–18], resourceful states in quantum metrology [19–21], ground states in condensed matter systems [22], the study of nonlocality [23], entanglement and Bell inequalities [24, 25]. Closely related to these developments, tools from artificial intelligence, genetic algorithms and competition have also been employed in chemistry, on the search for new pathways to organic molecules and closed autocatalytic reaction sets [26], the efficient training of neural networks for image classification [27] and the evolution of deep learning algorithms through competition [28]. Here, we propose harvesting some of these tools and ideas within the context of quantum computing and quantum algorithms.

Hilbert space is large [29–31] and – similarly to the vast and complex landscape of the biosphere – evolution may offer an efficient path to navigate its complexity. To test this idea, we apply genetic algorithms (GA) to the search for quantum circuits. As much as these ideas could benefit from a full scale quantum computer, there are not many such devices readily available for use at the present time [32–35]. We therefore focus on stabilizer circuits [36], which are efficiently simulable on classical computers [37, 38]. The stabilizer formalism is the natural language of quantum error correction [36], leading us to evolve quantum error correction codes (QECCs) [39].

Within the framework of stabilizer QECCs, we will demonstrate that evolution, given the appropriate fitness landscape, can successfully produce known examples of error correcting codes, notably the *Perfect* 5-qubit code [40], Shor's 9-qubit code [41], the 7-qubit *color* code [42, 43], and new QECCs. These are arguably simple textbook examples but, as we will point out, the sample space (modulo equivalences) of circuits in which such codes live is so large that random search becomes prohibitive, thus proving the principle that evolution efficiently drives the search.

Artificial selection might offer valuable opportunities in the current era of NISQ devices [44] in which elementary quantum gates are costly and noisy. Generically, random stabilizer circuits are capable of generating good codewords for QECCs [45], but evolution can go beyond typicality in guiding the search for *simple*, low-depth circuits more amenable to noisy devices. Device specificity can also be taken into account by devising fitness landscapes in terms of the complexity geometry metric [46], which penalizes gates according to hardware nuances [47].

At present, our artificial selection algorithm can be compared to the evolution of simple bacteria in controlled laboratory conditions where the fitness landscape is simple and well understood [48]. With more complex fitness functions and the addition of quantum hardware we anticipate improvements and systematic means of devising complex quantum circuits in various applications beyond stabilizer circuits and QECCs including but not limited to learning unitaries [49], quantum compiling [50–52], and hardware specific tailor-made circuits [32, 47]. We highlight that while evolution may be complementary to known circuit optimization schemes [51], its main strength relies on the possibility of creating novel and creative quantum circuits. All scripts made for this dissertation are available in the GitHub repository [53].

Dissertation organization

This dissertation is organized as follows. In Chapter 2, we make an introduction to the stabilizer formalism. Stabilizers are the natural language of QECCs. Accordingly, a basic understanding of its principles becomes essential for the discussions present in this work. The chapter covers all the material needed to understand the arguments used in the following chapters and provides useful algorithms. Chapter 3 presents the central theory of stabilizer QECCs providing the foundation for establishing a proper fitness function to drive a GA in search of suitable error correction circuits. Chapter 4 ties up the concepts presented in the previous chapters by conceiving a GA made to evolve quantum circuits. The chapter begins by introducing the idea of evolutionary algorithms in its general terms. Next, we show how to adapt it to the context of Clifford circuits. Section 4.2.3 is dedicated to a simple application of the GA, demonstrating its capabilities within a well-understood context. Following, we apply the tools of evolution to the search for QECCs in Section 4.3. Chapter 5 concludes with a brief discussion and an outlook for future developments and applications.

Before reading the main body of the dissertation, we invite the reader to read Appendices B to D to familiarize themselves with the notation used throughout this work and to review key results.

2
Stabilizer formalism

The stabilizers formalism was initially conceived to describe QECCs [36]. Stabilizers explore concepts of group theory by making the most of the beautiful idea that we can represent objects by their symmetry group [10, 36]. We will see that this leads us to define a class of quantum states which we can simulate in polynomial time on classical computers — this constitutes the famous Gottesman-Knill theorem [36]. Working with this class of states is fundamental for this work, as we do not yet have full access to quantum computers. Moreover, it defined our focus on the evolution of QECCs.

This chapter is divided as follows: in Section 2.1, we lay down the basics of stabilizer formalism, covering enough ground to give the reader sufficient knowledge to keep up with the discussions of the successive chapters; Section 2.2 introduces the concept of the tableau representation of stabilizer states, a popular way to represent and encode them; Section 2.3 presents three useful stabilizer algorithms. Additionally, Appendix E provides a summary and brings additional properties used in the chapter.

2.1
Stabilizer basics

In the stabilizer formalism, one represents quantum states by a unique set of operators, called *stabilizer* operators, instead of its decomposition into states of a given base. As we shall see, such representation allows simulation of a particular class of quantum states in a classical computer in polynomial time. Given a state $|\psi\rangle$, a operator U is a stabilizer of $|\psi\rangle$ if

$$U|\psi\rangle = +|\psi\rangle\,, \tag{2-1}$$

i.e., if $|\psi\rangle$ is an eigenstate of U with eigenvalue $+1$. As an illustration, the stabilizer operators of the Bell state

$$\left|\Phi^-\right\rangle = \frac{|00\rangle - |11\rangle}{\sqrt{2}} \tag{2-2}$$

are $\{I, -X_1X_2, Z_1Z_2, Y_1Y_2\}$. Note that the global phase of the operator is of relevance. The identity is a trivial stabilizer to any state and $-I$ stabilizes only the null state.

Define $\mathcal{S}(|\psi\rangle)$ as the set of stabilizers of $|\psi\rangle$. Under the operation of multiplication, $\mathcal{S}(|\psi\rangle)$ is a group. Here is a proof: let U and G stabilize $|\psi\rangle$, then

$$UG\,|\psi\rangle = (+1)^2\,|\psi\rangle = +\,|\psi\rangle\,. \tag{2-3}$$

$S(|\psi\rangle)$ is closed under multiplication. If U stabilizes $|\psi\rangle$, then so does U^{-1},

$$U^{-1}U\,|\psi\rangle = |\psi\rangle = U^{-1}\,|\psi\rangle\,. \tag{2-4}$$

Lastly, $I \in \mathcal{S}(|\psi\rangle)$ and operator multiplication is an associative. Q.E.D. Moreover, $\mathcal{S}(|\psi\rangle)$ is an Abelian group since if U and G stabilize $|\psi\rangle$, then

$$UG\,|\psi\rangle = GU\,|\psi\rangle \Leftrightarrow (UG - GU)\,|\psi\rangle = 0, \tag{2-5}$$

hence $[U, G] = 0$.

A crucial fact is that if, and only if, $|\psi\rangle = |\phi\rangle$, then $\mathcal{S}(|\psi\rangle) = \mathcal{S}(|\phi\rangle)$ [37]. This leads to the cornerstone idea of the stabilizers: represent a quantum state not by a vector in Hilbert space but by its stabilizer group. This is a common idea in group theory: looking at the set of transformations that leave an object invariant is the same as looking at the object itself. In the case of quantum stabilizers the object is a ray (one dimensional subspace) in Hilbert space.

It is not obvious that representing quantum states by their stabilizer group is better in any sense then considering the states directly. We need two more pieces of information to define a class of quantum states for which the stabilizer representation has an advantage in a computational sense. First, let $|\psi\rangle$ be a n-qubit state. An important property is that [36]

$$|\mathcal{S}(|\psi\rangle)| = 2^n. \tag{2-6}$$

The number of stabilizers grows exponentially with n. However, by group theory, a finite group $\mathcal{G}$ is spawned by precisely $\log_2|\mathcal{G}|$ elements called *generators* of the group [10]. Therefore, one may represent $\mathcal{S}(|\psi\rangle)$ by a subset of n generators, which grows linearly with n. Let $\langle\mathcal{S}(|\psi\rangle)\rangle$ be a set of generators of $\mathcal{S}(|\psi\rangle)$. For example, a possible generator set for the Bell state $|\Phi^-\rangle$ is $\langle\mathcal{S}(|\Phi^-\rangle)\rangle = \{-X_1X_2, Z_1Z_2\}$. Now suppose we have an initial quantum state $|\psi_0\rangle$ that evolves according to a quantum computation U. If G stabilizes $|\psi_0\rangle$, then $UGU^\dagger$ stabilizes $U\,|\psi_0\rangle$ since

$$(UGU^\dagger)U\,|\psi_0\rangle = U\,|\psi_0\rangle\,. \tag{2-7}$$

One can thus keep track of the transformations a state goes through by evaluating how it stabilizers transform under the conjugation $U \cdot U^\dagger$. Additionally, only n stabilizers need to be evaluated since $\mathcal{S}(U\,|\psi_0\rangle)$ can be defined by n generators. This concludes the first piece of information necessary for showing

the computational advantage offered by stabilizers.

Second, there is a particular class of quantum gates that have a simple action upon conjugation with Pauli operators: in particular, single spin Pauli operators are mapped to products of Pauli operators. Consider a circuit composed only of CNOTs, H, and P gates. Such circuits are called *Clifford* circuits or stabilizer circuits [54]. For example, consider how the Pauli operators transforms under conjugation with the H gate. Using $H = (X + Z)/2$ and employing the multiplication properties of Pauli letters, it is straightforward to check that

$$\begin{aligned} HXH^\dagger &= Z \\ HYH^\dagger &= -Y \\ HZH^\dagger &= X. \end{aligned} \tag{2-8}$$

Table 2.1 is a comprehensive list of the outcomes of conjugation of Pauli operators by Clifford gates. Since the CNOT gate is a 2-qubit gate, we must consider all two Pauli letters combinations. We can make use of this property about Clifford gates by exploiting the fact that any Pauli letter can be written as $i^p X^{u_x} Z^{u_z}$ in the following way: given an arbitrary n-qubit Pauli operator $g \in \mathcal{G}_n$, it can be expressed as

$$g = i^p X_1^{u_{x_1}} Z_1^{u_{z_1}} \dots X_n^{u_{x_n}} Z_n^{u_{z_n}}, \tag{2-9}$$

where $\vec{u} = (u_{x_1}, u_{z_1}, \dots, u_{x_n}, u_{z_n})$ is vector with $2n$ binary entries since Pauli operators square to identity, and $p = 0, 1, 2, 3$. Thus, we can represent g by the set $\{\vec{u}, p\}$. Now, since Pauli operators conjugate to single product of Pauli operators under the action of Clifford gates, a set $\{\vec{u}, p\}$ is mapped to $\{\vec{u'}, p'\}$ under conjugation with a Clifford circuit. The action of a Clifford gate then corresponds to a dynamics defined in terms of simple local operations acting on $\{\vec{u}, p\}$. This is the second piece of information.

We now show that Clifford circuits produce states encodable by $2n^2 + n$ bits. Consider a n-qubit initial state $|\psi_0\rangle = \bigotimes_{i=1}^n |a_i\rangle_i$ with a_i equal to 0 or 1, i.e., $|\psi_0\rangle$ is a state of the computational basis. As discussed, we can represent $|\psi_0\rangle$ as the set $\langle \mathcal{S}(|\psi_0\rangle)\rangle$ with a particular easy set being $\langle \mathcal{S}(|\psi_0\rangle)\rangle = \{\pm Z_1, \pm Z_2, \dots, \pm Z_n\}$ (the sign depends on the i-th qubit being $|0\rangle$ or $|1\rangle$). Applying a Clifford circuit C to $|\psi_0\rangle$ maps each generator $\pm Z_i$ to a simple product of Pauli letters encoded by $\{\vec{u}_i, p_i\}$. Moreover, by Table 2.1, each p_i can only assume the values of 0 or 2 (corresponding to the global phases of ± 1) after the application of C. Hence, one can represent $|\psi\rangle = C|\psi_0\rangle$ by the set $\{\{\vec{u}_i, p_i\}_i\}$, where each vector $\vec{u}_i$ have $2n$ binary entries and each p_i is also

Table 2.1: Conjugation of Pauli operators under the action of Clifford gates.

Gate	Pauli operator	Conjugation output
H	X	Z
	Y	$-Y$
	Z	X
P	X	Y
	Y	$-X$
	Z	Z
CNOT	X_1	X_1X_2
	X_2	X_2
	Z_1	Z_1
	Z_2	Z_1Z2
	Y_1	Y_1X_2
	Y_2	Z_1Y_2
	X_1X_2	X_1
	X_1Y_2	Y_1Z_2
	X_1Z_2	$-Y_1Y_2$
	Y_1X_2	Y_1
	Y_1Y_2	$-X_1Z_2$
	Y_1Z_2	X_1Y_2
	Z_1X_2	Z_1X_2
	Z_1Y_2	Y_2
	Z_1Z_2	Z_2

binary. Since $|\{\{\vec{u}_i, p_i\}_i\}| = |\langle\mathcal{S}(|\psi_0\rangle)\rangle| = n$, we can describe $|\psi\rangle$ by $2n^2 + n$ bits of information.

This is the essence of the *Gottesman–Knill theorem* [36], which states that a n-qubit quantum state, initialized in some state of the computational basis and prepared with a Clifford circuit, can be simulated efficiently on a classical computer. We call such states *stabilizer states*.

2.2 Tableau representation

A widespread form to represent a stabilizer state is by building its *tableau*. For example, consider the n-qubit state $|0\rangle^{\otimes n}$ which we shall always represent by the generator set $\{Z_1, Z_2, \ldots, Z_n\}$, unless otherwise stated. We construct its tableau by stacking the generators on the rows of a $n \times n + 1$ matrix:

$$\left|\begin{array}{ccccc} Z & I & \ldots & I & +1 \\ I & Z & \ldots & I & +1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ I & I & \ldots & Z & +1 \end{array}\right|, \tag{2-10}$$

The first n columns are the Pauli letters that constitute each generator, the

index of the column being the index of the qubit the operator acts upon, and the last column stores the overall phase. An equivalent form is to use the binary $\{\vec{u}, p\}$ representation

$$\left|\begin{matrix} 01 & 00 & \dots & 00 & 0 \\ 00 & 01 & \dots & 00 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 00 & 00 & \dots & 01 & 0 \end{matrix}\right|, \tag{2-11}$$

This representation is suitable for calculations since it simplifies multiplications between Pauli operators as follows: given $\{\vec{u}, p\}$ and $\{\vec{u}', p'\}$,

$$\{\vec{u}, p\} \cdot \{\vec{u}', p'\} = \{\vec{u} + \vec{u}', p + p'\}, \tag{2-12}$$

where the sums $\vec{u}+\vec{u}'$ and $p+p'$ are bitwise modulo 2 and 4 sums, respectively. We call respectively (2-10) and (2-11) the literal and binary tableau representations. They may be used interchangeably, being the literal easier to read and interpret, and the binary more suited for performing computations.

The tableau representation, especially the binary, is used to encode and compute the evolution of stabilizer states. Given a Clifford circuit, Table 2.1 is used to update the rows of the literal tableau. Conversely, we can express the conjugation transformations by simple rules relating the components of $\{\vec{u}_i, p_i\} \in \langle \mathcal{S}(|\psi\rangle)\rangle$ and update the binary tableau. Table 2.2 lists the transformation rules for the binary form.

Table 2.2: Transformation rules for the stabilizer binary form. The sums for the components of $\vec{u}$ and p are respectively modulo 2 and 4. For the CNOT, indices i and j are respectively the control and the target qubits.

Gate	Rules
H	$u_{x_i} \leftrightarrow u_{z_i}$ if $(u_{x_i}, u_{z_i}) = (1,1)$, $p = p + 2$
P	$u_{z_i} \leftrightarrow u_{z_i} + u_{z_i}$ if $(u_{x_i}, u_{z_i}) = (1,1)$, $p = p + 2$
CNOT	$u_{x_j} \rightarrow u_{x_j} + u_{x_i}$ $u_{z_i} \rightarrow u_{z_i} + u_{z_j}$ if $(u_{x_i}, u_{z_i}) = (1,1)$ and $(u_{x_j}, u_{z_j}) = (1,1)$, $p = p + 2$ if $(u_{x_i}, u_{z_i}) = (1,0)$ and $(u_{x_j}, u_{z_j}) = (0,1)$, $p = p + 2$

As an illustrative simple example, consider the circuit depicted in Figure 2.1. Starting from the tableau for the initial state $|00\rangle$,

$$\left|\begin{matrix} Z & I & +1 \\ I & Z & +1 \end{matrix}\right|, \tag{2-13}$$

we gate-by-gate evaluate the evolution. The circuit is composed by a H_1 followed by a CNOT gate with qubit 1 and 2 as the control and target,

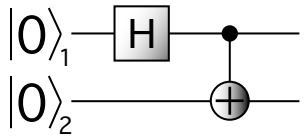

Figure 2.1: Quantum circuit example.

respectively. The general procedure is as follows: in the order of application of the gates, we update the rows of the tableau according to the proper transformation listed in Table 2.1 (or Table 2.2 if working with the binary tableau). The H_1 maps $Z_1 \mapsto X_1$ (note that identities are always trivially mapped to identities). The CNOT maps $X_1 \mapsto X_1X_2$ and $Z_2 \mapsto Z_1Z_2$. Hence,

$$\begin{vmatrix} Z & I & +1 \\ I & Z & +1 \end{vmatrix} \xrightarrow{H_1} \begin{vmatrix} X & I & +1 \\ I & Z & +1 \end{vmatrix} \xrightarrow{\text{CNOT}} \begin{vmatrix} X & X & +1 \\ Z & Z & +1 \end{vmatrix}. \tag{2-14}$$

It is straightforward to check that the final state is the Bell state $|00\rangle+|11\rangle/\sqrt{2}$, where $\{+X_1X_2, +Z_1Z_2\}$ is indeed a proper set of stabilizer generators.

2.2.1 Row-reduced echelon form

Stabilizer states are uniquely represented by their stabilizer group. However, there are multiple sets of generators that spawns the same group. For example, consider the last tableau of Equation (2-14). All the following are equivalents to it:

$$\mathcal{T}_1 = \begin{vmatrix} X & X & +1 \\ Z & Z & +1 \end{vmatrix}, \mathcal{T}_2 = \begin{vmatrix} X & X & +1 \\ Y & Y & -1 \end{vmatrix}, \mathcal{T}_3 = \begin{vmatrix} Y & Y & -1 \\ X & X & +1 \end{vmatrix}. \tag{2-15}$$

Since the multiplication of two stabilizers is a stabilizer, one can construct equivalent tableaux by multiplying different rows (multiplying a row by itself gives the identity operator, which renders no information about the state). Row transposition, swapping of two rows, is also a valid operation that leaves the stabilizer structure unaltered. Hence, $\mathcal{T}_2$ is built by left multiplying the first and second row of $\mathcal{T}_1$, and $\mathcal{T}_3$ by swapping the rows of $\mathcal{T}_2$.

One can thus use the elementary operations of row multiplication and transposition to rearrange a tableau to achieve a desirable form without changing the underlying represented state. This motivates the creation of an algorithm along the lines of the Gauss-Jordan elimination to arrange the tableau in a unique *row-reduced echelon* form, also known as *canonical* form. Besides defining a unique form for the tableau, the canonical form will be required for the inner-product algorithm that we present in the following section. We closely follow [55] to describe an algorithm that takes an arbitrary

tableau and transforms it, via row operations, into the row-reduced echelon form illustrated in Figure 2.2.

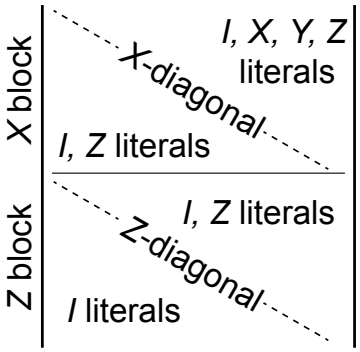

Figure 2.2: Row-reduced echelon (canonical) form of a general tableau. The tableau is divided in two blocks. The top block (the X-block) contains only X literals on the principal diagonal, only I and Z below it, and I, X, Y and Z above it. The bottom block (the Z-block) contains only Z literals on the principal diagonal, only I below it, and only I and Z above it. This construction minimizes the number of non-I literals in each row.

The algorithm works progressively to arrange the tableau in a form that contains a set of generators with the minimum number of X and Y literals at the top and a set with the minimum number of Z literals at the bottom. The rationale is strikingly similar to Gauss elimination: by exploiting the facts that Pauli letters square to unity and $XY \propto Z$, the leading non-I literal of each row is used to transform, via row multiplication, the literals on the same column into I, Z or X in a clever manner. Let $\mathcal{T}$ be a n-qubit $n \times n+1$ tableau, and let $i, j \in \{1, \ldots, n\}$ indices that will run through the rows and columns of $\mathcal{T}$ respectively. Initialize $i = j = 1$. The following steps arrange $\mathcal{T}$ in the canonical form:

1. Search for the lowest index k such that $\mathcal{T}_{kj} = X$ or Y;

   (a) If k exists: (i) swap rows i and k, (ii) left multiply row i with all rows with index different from i, (iii) sum 1 to i and j, (iv) if i equals to n, break the loop, else if j equals to n, go to step 2 and assign $j = 1$, else go back to step 1;

   (b) If k does not exist: (i) sum 1 to j, (ii) if j equals to n, go to step 2 and assign $j = 1$, else go back to step 1.

2. Search for the lowest index k such that $\mathcal{T}_{kj} = Z$;

   (a) If k exists: (i) swap rows i and k, (ii) left multiply row i with all rows with index different from i and literal in the j-th column equal

to X or Z, (iii) sum 1 to i and j, (iv) if i or j equals to n, break the loop, else go back to step 2;

(b) If k does not exist: (i) sum 1 to j, (ii) if j equals to n, break the loop, else go back to step 2.

Algorithm 1 presents the pseudocode that executes the steps detailed above.

2.3 Stabilizer algorithms

Stabilizer states are interesting because one can simulate them efficiently with a classical computer. However, it is unclear how we can calculate specific quantities important to quantum mechanics from the tableau representation. For example, given the tableaux of two different states, how does one evaluate the inner product between them? Fortunately, we can ingeniously build algorithms capable of performing such calculations. We thus end this chapter by presenting three valuable algorithms concerning stabilizers states: the inner-product, measurement, and the von Neumann entropy.

2.3.1 Inner-product

One can efficiently evaluate the inner-product between two stabilizer states using their tableau representation [55]. First, define the *basis* form as the tableau with the following structure:

$$\begin{vmatrix} Z & I & \dots & I & \pm 1 \\ I & Z & \dots & I & \pm 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ I & I & \dots & Z & \pm 1 \end{vmatrix}. \tag{2-16}$$

The basis form is just the tableau of an arbitrary n-qubit state of the computational basis. Note that it is already in the canonical form.

Let $|\psi\rangle$ and $|\phi\rangle$ be two n-qubit stabilizers states where, without loss, $|\psi\rangle$ is represented by a basis form tableau $\mathcal{T}_{|\psi\rangle}$ and $|\phi\rangle$ is an arbitrary stabilizer state with $\mathcal{T}_{|\phi\rangle}$ in its canonical form. Since the X-block of $\mathcal{T}_{|\psi\rangle}$ is empty, all stabilizers in $\mathcal{S}(|\psi\rangle)$ contains only I and Z Pauli letters. Additionally, all stabilizers of $\mathcal{S}(|\phi\rangle)$ that contains only I and Z literals are necessarily spawned by multiplications between the rows of the Z-block of $\mathcal{T}_{|\phi\rangle}$ (multiplying a row of the X-block with any other row will necessarily have at least one X or Y literal due to the X-block structure). By property 7 listed in Appendix E, it suffices to inspect if there is a row in the Z-block of $\mathcal{T}_{|\phi\rangle}$ equal to a row in $\mathcal{T}_{|\psi\rangle}$, but with different overall sign, to assess if $\langle\psi|\phi\rangle = 0$. Suppose such condition

Algorithm 1 Tableau canonical transformation

Input: A $n \times n+1$ tableau $\mathcal{T}$
Output: $\mathcal{T}$ in row-reduced echelon form
$\Rightarrow$ rows($\mathcal{T}$) gets the number of rows of $\mathcal{T}$
$\Rightarrow$ rowSwap($\mathcal{T}$, i, j) swaps the i-th and j-th rows of $\mathcal{T}$
$\Rightarrow$ rowMult($\mathcal{T}$, i, j) multiplies the i-th and j-th rows of $\mathcal{T}$

```
1: i = 1
2: j = 1
3: k = i
4: n = rows(T)
5: while i ≠ n or j ≠ n do                          ▷ X-block formation
6:     if T_kj = X or Y then    ▷ search row with leading X or Y on j-th col.
7:         rowSwap(T, i, k)
8:         for l ∈ {1, ..., n} do
9:             if l ≠ i then
10:                rowMult(T, i, l)
11:            end if
12:        end for
13:        i = i + 1                                 ▷ reduces sub-matrix
14:        j = j + 1
15:        k = i
16:    else
17:        k = k + 1
18:        if k = n then
19:            j = j + 1          ▷ search failed for all rows, advance column
20:            k = i
21:        end if
22:    end if
23: end while
24: j = 1
25: while i ≠ n or j ≠ n do                          ▷ Z-block formation
26:    if T_kj = Z then          ▷ search row with leading Z on j-th column
27:        rowSwap(T, i, k)
28:        for l ∈ {1, ..., n} do
29:            if l ≠ i and T_lj = Z or Y then
30:                rowMult(T, i, l)
31:            end if
32:        end for
33:        i = i + 1                                 ▷ reduces sub-matrix
34:        j = j + 1
35:        k = i
36:    else
37:        k = k + 1
38:        if k = n then
39:            j = j + 1          ▷ search failed for all rows, advance column
40:            k = i
41:        end if
42:    end if
43: end while
```

is not met. We claim that $\langle\psi|\phi\rangle = 2^{-s/2}$, where s is the number of rows on the X-block of $\mathcal{T}_{|\phi\rangle}$. The reason goes as follows. Each row of the X-block of $\mathcal{T}_{|\phi\rangle}$, with a leading X literal at some column with index j, indicates that the j-th qubit of $|\phi\rangle$ must be in an unbiased superposition of $|0\rangle$ and $|1\rangle$. Since the j-th qubit of $|\psi\rangle$ is either $|0\rangle$ or $|1\rangle$ by construction, it follows that these qubits contribute with a factor of $1/\sqrt{2}$ to the inner-product.

Now, given two arbitrary n-qubit stabilizers states $|\psi\rangle$ and $|\phi\rangle$, we extend the above reasoning to evaluate $\langle\psi|\phi\rangle$. Let C be a Clifford circuit such that $C|0\rangle^{\otimes n} = |\psi\rangle$. We may write the inner-product as

$$\langle\psi|\phi\rangle = \langle\psi| C^{-1}C |\phi\rangle = \langle 0|^{\otimes n} C |\phi\rangle . \tag{2-17}$$

Since $|0\rangle^{\otimes n}$ has a tableau in the basis form and one can evaluate the canonical form of $\mathcal{T}_{C|\phi\rangle}$ with Algorithm 1, one has all the necessary ingredients to calculate $\langle\psi|\phi\rangle$. Note that we can reverse the roles of $|\psi\rangle$ and $|\phi\rangle$ in the above argument.

A crucial element of the inner-product computation is access to the Clifford circuit that synthesizes one of the states. We assume that the circuits are always available since we do not work directly with random states (we generate random stabilizer states via known random circuits applied to $|0\rangle^{\otimes n}$). Nevertheless, given an arbitrary stabilizer state $|\psi\rangle$ with no circuit (just its tableau, for example), there are algorithms capable of constructing a Clifford circuit C such that $C|0\rangle^{\otimes n} = |\psi\rangle$ [55].

Algorithm 2 computes the inner-product of two n-qubit stabilizer states $|\psi\rangle$ and $|\phi\rangle$, for which we assume that we have access to a Clifford circuit $C_{|\psi\rangle}$ such that $C_{|\psi\rangle}|0\rangle^{\otimes n} = |\psi\rangle$.

2.3.2 Measurement

The stabilizer formalism also admits simulation of one-qubit measurements on the computational basis [54]. By property 10 of Appendix E, the j-th qubit of a stabilizer state can only be either $|0\rangle(|1\rangle)$ or in an unbiased superposition of $|0\rangle$ and $|1\rangle$. One can verify which one is the case by searching for X/Y literals on the j-th column of the canonical tableau of the state. If there are such literals, the qubit must be in an unbiased superposition, and the measurement will have a *random* outcome of 0 or 1 with equal probability. One then uses a random source to decide the outcome. Else, the outcome is *deterministic* with the sign of generator with leading Z literal in the j-th column determining the outcome. For the latter case, there is no need to update the tableau to compute the post-measurement state since it did not collapse. For

Algorithm 2 Stabilizer inner-product

Input: (i) A $n \times n+1$ tableau $\mathcal{T}_{|\psi\rangle}$ for $|\psi\rangle$, (ii) a $n \times n+1$ tableau $\mathcal{T}_{|\phi\rangle}$ for $|\phi\rangle$, (iii) Clifford circuit $C_{|\psi\rangle}$ for $|\psi\rangle$, (iv) basis form tableau $\mathcal{T}_{|0\rangle^{\otimes n}}$ of $|0\rangle^{\otimes n}$

Output: Inner product of $|\psi\rangle$ and $|\phi\rangle$

$\Rightarrow$ rows($\mathcal{T}$) gets the number of rows of $\mathcal{T}$

$\Rightarrow$ canon($\mathcal{T}$) brings tableau $\mathcal{T}$ to its canonical form

$\Rightarrow$ conj($\mathcal{T}$, C) conjugates tableau $\mathcal{T}$ with the Clifford circuit C

$\Rightarrow$ mult($\mathcal{T}$, i, $\mathcal{T}'$, j) multiplies the i-th and j-th rows of tableaux $\mathcal{T}$ and $\mathcal{T}'$

$n = \text{rows}(\mathcal{T}_{|\psi\rangle})$
$\mathcal{T}_{|\phi\rangle} = \text{conj}(\mathcal{T}_{|\phi\rangle}, \text{C})$
$\mathcal{T}_{|\phi\rangle} = \text{canon}(\mathcal{T}_{|\phi\rangle})$
$s = 0$
$i = 1$
while $i \neq n$ **do**
    $j = 1$
    while $j \neq n$ **do**
        if $\mathcal{T}_{|\phi\rangle,ij} = X$ or Y **then**
            $s = s + 1$
            $i = i + 1$
            $j = n$
        else
            $j = j + 1$
            if $j = n$ **then** ▷ it is a row of the Z-block
                $i = i + 1$
                for $k \in \{1, \ldots, n\}$ **do** ▷ checking orthogonality
                    if mult($\mathcal{T}_{|\phi\rangle}$, i, $\mathcal{T}_{|0\rangle^{\otimes n}}$, k) $= -$I **then**
                        return 0
                    end if
                end for
            end if
        end if
    end while
end while
return $2^{-s/2}$

the former, the j-th qubit collapsed into $|0\rangle$ or $|1\rangle$ depending on the result of the random source. Thus, one updates the tableau by replacing the row with the X/Y literals on the j-th by $\pm Z_j$.

Given a canonical tableau $\mathcal{T}$ and a random source, Algorithm 3 displays the pseudocode for measuring the j-th qubit in the computational basis giving, as output, the measurement outcome and the post-measurement tableau.

2.3.3 Entropy of a subsystem

By property 11 of Appendix E, the density operator ρ of a n-qubit stabilizer state $|\psi\rangle$ can be expressed as

$$\rho = \frac{1}{2^n} \sum_{s \in \mathcal{S}(|\psi\rangle)} s. \tag{2-18}$$

If we imagine that the qubits are laid in an array, tracing out a region B yields the state,

$$\rho_A = \frac{1}{2^n} \sum_{s \in \mathcal{S}(|\psi\rangle)} \mathrm{Tr}_B(s). \tag{2-19}$$

It turns out the von Neumann entropy of a region A is given by

$$S_A = \mathcal{I}_A - |A| \tag{2-20}$$

where $\mathcal{I}_A$ is the number of independent stabilizers (arithmetic modulo 2) when restricted to region A, and $|A|$ the number of qubits in A [56].

In practice, what needs to be done to obtain the von Neumann entropy is the following:

1. For a set of generators $\langle \mathcal{S}(|\psi\rangle)\rangle$ written in the binary form $\vec{u}_i$, build a binary matrix $[\vec{u}_i]$ by stacking $\vec{u}_i$ into its rows (the overall signs p_i are not important);

2. Delete each column of the matrix that corresponds to qubits outside region A. Denote the resulting matrix $[\vec{u}_i]_A$.

3. Evaluate the rank of $[\vec{u}_i]_A$ modulo 2. This rank is $\mathcal{I}_A$.

Algorithm 4 shows the pseudocode for the evaluation of the von Neumann entropy.

Algorithm 3 Stabilizer measurement

Input: (i) A $n \times n+1$ tableau $\mathcal{T}$, (ii) index j of the qubit to be measured
Output: (i) output m of measurement, (ii) post-measurement tableau $\mathcal{T}$
$\Rightarrow$ rows($\mathcal{T}$) gets the number of rows of $\mathcal{T}$
$\Rightarrow$ canon($\mathcal{T}$) brings tableau $\mathcal{T}$ to its canonical form
$\Rightarrow$ random() outputs 0 or 1 with an equal probability of 0.5

$\mathcal{T} = \text{canon}(\mathcal{T})$
$i = 1$
$n = \text{rows}(\mathcal{T})$
while $i \neq n$ **do**
    if $\mathcal{T}_{ij} = X$ or Y **then** ▷ random outcome
        $m = \text{random}()$ ▷ drawing outcome
        for $k \in \{1, \ldots, n\}$ **do** ▷ updating $\mathcal{T}$
            if $k \neq j$ **then**
                $\mathcal{T}_{ik} = I$
            else
                $\mathcal{T}_{ik} = Z$
            end if
        end for
        if $m = 0$ **then** ▷ updating phase according to outcome m
            $\mathcal{T}_{i,n+1} = +1$
        else
            $\mathcal{T}_{i,n+1} = -1$
        end if
        $i = n$
    else if $\mathcal{T}_{ij} = Z$ **then** ▷ deterministic outcome
        if $\mathcal{T}_{i,n+1} = +1$ **then** ▷ getting output from phase of the operator
            $m = 0$
        else
            $m = 1$
        end if
        $i = n$
    else
        $i = i + 1$
    end if
end while
return m, $\mathcal{T}$

Algorithm 4 von Neumann entropy

Input: (i) A $n \times n+1$ tableau $\mathcal{T}$ in binary form, (ii) indices $i \in \mathcal{A}$ of qubits in region A

Output: (i) The von Neumann entropy S_A of region A

$\Rightarrow$ rows($\mathcal{T}$) gets the number of rows of $\mathcal{T}$

$\Rightarrow$ delCol(T, i) deletes the i-th column of matrix T

$\Rightarrow$ rankMod2(T) outputs the rank modulo 2 of a binary matrix T

1: $n = \text{rows}(\mathcal{T})$
2: $T = \text{delCol}(\mathcal{T}, \text{n}+1)$ ▷ deletes phase column
3: $\mathcal{I}_A = n$
4: **for** $i \in \{1, \ldots, n\}$ **do**
5: **if** $i \notin \mathcal{A}$ **then**
6: $T = \text{delCol}(\mathcal{T}, \text{i})$
7: $\mathcal{I}_A = \mathcal{I}_A - 1$
8: **end if**
9: **end for**
10: $S_A = \mathcal{I}_A - \text{rankMod2(T)}$
11: **return** S_A

3
Quantum error correction codes

Quantum computation relies on the precise control of qubits to generate entanglement of their degrees of freedom. Unfortunately, the physical systems proposed to build a quantum computer rely on systems that are highly prone to decoherence due to interactions with the external environment. Consequently, preserving the quantum coherence between its parts becomes imperative to render the computation reliable, which lead to the development of QECCs.

The field of quantum error correction is vast and spawned many subfields in the last couple of decades. This chapter aims to provide the basic knowledge to stabilizer QECCs, the class of quantum error correction we seek to evolve with GAs. This chapter is organized as follows. Section 3.1 introduces error correction, starting from the classical counterpart and rapidly moving to the quantum paradigm with its problematics. Section 3.2 formalizes the theory by constructing a general scheme for stabilizer QECCs. Sections 3.2.2 and 3.2.3 end the chapter by providing two examples: Shor's 9-qubit and the perfect 5-qubit code.

3.1
Introduction to error correction

We start by presenting fundamental notions of classical error correction using the 3-bit code as an example. Moving to the quantum counterpart, quantum mechanics poses three significant problems at the onset when we try to implement the classical scheme. By exploring the 3-qubit code, we demonstrate how to overcome each challenge by exploiting quantum mechanical features.

3.1.1
Classical error correction fundamentals

For simplicity, consider data encoded in strings of bits of the alphabet $\mathcal{A} = \{0, 1\}$. Given an arbitrary string at an initial time, we desire to retrieve the information stored in it at a future time, and perhaps a different location, even if the data have been damaged. Thus, we need to protect it in some form. The simplest method of protection is by adding redundancy by mere repetition of the data content. Humans use this strategy in oral communication all the

time, asking for a speaker to repeat himself when a pronounced sentence is not understood. Hearing a phrase multiple times allows one to reconstitute it by piecing it together.

Consider then the simple task of transmitting the bit 1 through a noisy channel. We model the classical noisy channel as a process where a bit has a probability p $(1-p)$ of flipping (not flipping)[1]. Instead of transmitting the raw data, we transmit an encoded version using the 3-bit code as follows:

$$\begin{aligned} 0 &\mapsto 000 \\ 1 &\mapsto 111. \end{aligned} \tag{3-1}$$

The above code maps the alphabet $\mathcal{A}$ into the encoded set $\mathcal{C} = \{000, 111\}$. The elements of $\mathcal{C}$ are called *codewords* of the error correction code. Suppose, after transmission of the encoded string 111, the first bit flipped making the receptor receive the damaged string 011. By applying a majority vote protocol, the recipient concludes correctly that the original data must have been 1.

Notice that the correction process fails whenever two or three errors occur. Consequently, we say that the 3-bit code can protect up to $t = 1$ errors. Additionally, we define the so-called *distance* of the code as the minimum number of errors required to flip a codeword into another one entirely. Clearly, the 3-bit code has distance $d = 3$. An equivalent form of defining the distance is the minimum error size that goes undetected, i.e., the receptor wrongly thinks no errors occurred. The size of correctable errors t and the distance d are related by [39]

$$d = 2t + 1. \tag{3-2}$$

Since d and t are positive integer numbers, a 2-bit code cannot correct any errors.

It is usual to classify a correction code by the list $[n, k, d]$ [10], where n is the number of bits of the codeword space, k the number of bits of the data alphabet, and d the distance — making the 3-bit code a $[3, 1, 3]$ code. An analogous classification scheme applies to quantum codes, however *"number of bits"* is properly replaced by the dimension of the given Hilbert spaces. Additionally, one uses double square-brackets to differentiate between classical and quantum codes.

The 3-bit code is an example of a repetition code [10]. Despite the existence of several other types of correction codes with increasing complexity, a common element to all of them is the use of redundancy, a feature also essential for quantum correction codes.

[1]By flipping, we mean 0 becoming 1 and vice-versa.

3.1.2 Towards a functional QECC

Quantum states replace the classical bits in quantum computation. In this work we mainly work with qubits (quantum correction schemes for continuous variables exist [57, 58]), states of two-level systems, since we intend to evolve stabilizer QECCs. Expressed in the computational basis $\mathcal{B} = \{|0\rangle, |1\rangle\}$, an arbitrary qubit state $|\psi\rangle$ may me written as

$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle , \tag{3-3}$$

where we require $|\alpha|^2 + |\beta|^2 = 1$ for normalization.

Quantum speedups over classical algorithms depend on the exploitation of superposition and entanglement [10]. Hence, in a quantum error correction scheme, we want to protect the coherence of a given state, i.e., the relative phases of its components. If we naively try to make a quantum version of the 3-bit code, we are immediately confronted with three particularities of quantum mechanics that we need to overcome. First (1), *errors are continuous*. To make the matter more evident, note that the normalization criterion allows the parameterization of a qubit state by two angles as

$$|\psi\rangle = \cos\tfrac{\theta}{2} |0\rangle + e^{i\phi} \sin\tfrac{\theta}{2} |1\rangle . \tag{3-4}$$

This parameterization leads to representing an arbitrary pure state as a point on the surface of a 2-sphere with unit radius — the so-called Bloch sphere [56] depicted in Figure 3.1. Thus, one can understand the action of a unitary operator U as translating a point on the Bloch sphere to another point with different parameterization angles, i.e,

$$U |\psi\rangle = \cos\tfrac{\theta+\Delta\theta}{2} |0\rangle + e^{i(\phi+\Delta\phi)} \sin\tfrac{\theta+\Delta\theta}{2} |1\rangle . \tag{3-5}$$

Since there are infinite points on the surface, the state may assume a continuum of values after an error occurs. Seemingly, one needs to detect and correct an infinite number of potential errors. This situation directly contrasts with the classical counterpart, where the flipping of a bit was the unique possible error.

Second (2), the no-cloning theorem excludes the possibility of making copies an arbitrary state [59]. We now show a simple proof of the no-cloning theorem: for the sake of contradiction, suppose there is a unitary U capable of cloning an arbitrary state $|\psi\rangle$ such that:

$$U |\psi\rangle |0\rangle = |\psi\rangle |\psi\rangle . \tag{3-6}$$

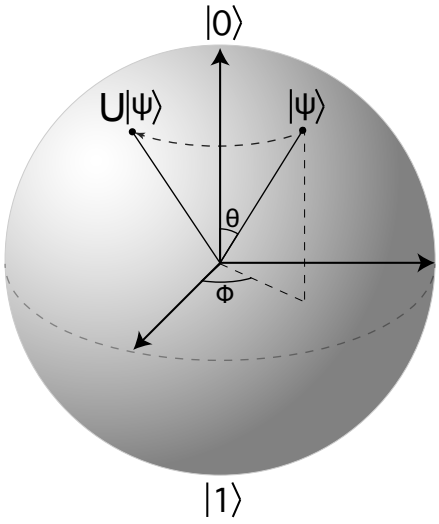

Figure 3.1: Geometrical representation of an arbitrary qubit state on a Bloch sphere. The state is defined by the two angles $\{\theta, \phi\}$, and rests on the surface of a 2-sphere. Unitary operators have the action of translating points on the surface.

Let $|\psi\rangle$ and $|\phi\rangle$ be arbitrary states. It follows that

$$\begin{aligned}(\langle\psi|\,\langle\psi|)(|\phi\rangle\,|\phi\rangle) &= \langle\psi|\phi\rangle^2 \\ &= (\langle\psi|\,\langle 0|\,U^\dagger)(U\,|\phi\rangle\,|0\rangle) \\ &= \langle\psi|\phi\rangle\,. \end{aligned} \tag{3-7}$$

The equality $\langle\psi|\phi\rangle^2 = \langle\psi|\phi\rangle$ is only satisfied if the states are orthogonal or the same, which contradicts the assumption that U clones arbitrary states. Q.E.D. The possibility to copy data was fundamental for the classical code — we introduced redundancy by making several copies of the data bit. One needs to devise an alternative way for adding redundancy to a quantum system.

Third (3), measurements lead to the collapse of quantum states, i.e., measurements destroy coherence. The majority vote scheme employed in the 3-bit code hinged on the possibility of comparing the received bit-string with the set of established codewords. The collapse of the wavefunction renders such a procedure meaningless since states in superposition would irreversibly collapse with information being lost. In some sense, the wavefunction collapse is a more fundamental issue than the no-cloning problem since even if it were possible to clone an arbitrary state, it would be of no use if one cannot measure it.

Fortunately, all three challenges have proper solutions. We now show how to circumvent each issue and, in the following sections, we develop a more general theory for quantum error correction concentrating on stabilizer codes.

For problem (1), it turns out that we can decompose an arbitrary error into a finite number of components, each of which corresponds to a unique error type. Let U be a unitary operator that acts on single-qubits. It follows that we can decompose U as

$$U = c_I I + c_X X + c_Y Y + c_Z Z \tag{3-8}$$

since $\{I, X, Y, Z\}$ spanned the space of operators acting on qubits [10]. We can then express $U|\psi\rangle$ as

$$U|\psi\rangle = c_I I|\psi\rangle + c_X X|\psi\rangle + c_{XZ} XZ|\psi\rangle + c_Z Z|\psi\rangle \tag{3-9}$$

since $Y \propto XZ$. Therefore, by performing an adequate projection of $U|\psi\rangle$ into one of the four components of Eq. (3-9), we are only required to correct two error types: X and Z [10, 36]. This passage from a continuous spectrum of errors to just two is called the *digitization* of quantum errors [10, 39]. The X operator is the quantum analog to the classical bit-flip, since

$$X(\alpha|0\rangle + \beta|1\rangle) = \alpha|1\rangle + \beta|0\rangle . \tag{3-10}$$

The Z operator has no classical counterpart, for its action is to give a relative phase to a state:

$$Z(\alpha|0\rangle + \beta|1\rangle) = \alpha|0\rangle - \beta|1\rangle . \tag{3-11}$$

For this reason, it is often referred to as a *phase-flip* error [39]. As we shall see, the Z error will prevent the quantum analog for the 3-bit code from correcting all possible single-qubit errors.

For problem (2), quantum *entanglement* is used to add redundancy to a system. In its essence, redundancy serves to spread the information one wishes to protect. By copying a bit, one distributes its information content among other bits, effectively decentralizing it. In a similar fashion, when a quantum system entangles with other subsystems, the information contained in its relative phases disperses into the more extensive system effectively introducing redundancy. Consider then the 3-qubit code defined by encoding $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ as

$$|\psi\rangle \mapsto |\psi\rangle_L = \alpha|000\rangle + \beta|111\rangle , \tag{3-12}$$

where the subscript L stands for *logical*. We say that the state was encoded into the logical state $|\psi\rangle_L$. It is straightforward to check that the circuit depicted in Figure 3.2 performs the encoding. The encoding circuit (EC) expands the original two-dimensional Hilbert space into an eight-dimensional one. Moreover, the relative phase is now belongs to a 3-qubit state. Actually, the logical state lives in a two-dimensional subspace $\mathcal{C}$, called *code space*,

spanned by the codewords states $\{|000\rangle, |111\rangle\}$. We may define the encoding as mapping the computational basis states into codewords:

$$|0\rangle \mapsto |000\rangle = |0\rangle_L \tag{3-13}$$

$$|1\rangle \mapsto |111\rangle = |1\rangle_L . \tag{3-14}$$

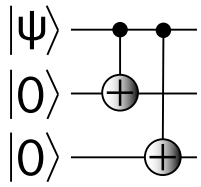

Figure 3.2: EC for the 3-qubit code. This circuit maps $\alpha|0\rangle + \beta|1\rangle$ into $\alpha|000\rangle + \beta|111\rangle$.

A critical remark is that codewords are not unique. The EC defines the code space with infinite possible orthogonal bases. For example, consider the single-qubit diagonal basis $\{|+\rangle, |-\rangle\}$ defined by

$$|+\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}, \; |-\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}. \tag{3-15}$$

In this base, $|\psi\rangle$ becomes

$$|\psi\rangle = \frac{\alpha+\beta}{\sqrt{2}}|+\rangle + \frac{\alpha-\beta}{\sqrt{2}}|-\rangle . \tag{3-16}$$

Consider again the EC of Figure 3.2. It maps the diagonal basis into GHZ states [60] as

$$|+\rangle \mapsto \frac{|000\rangle + |111\rangle}{\sqrt{2}} = |+\rangle_L \tag{3-17}$$

$$|-\rangle \mapsto \frac{|000\rangle - |111\rangle}{\sqrt{2}} = |-\rangle_L , \tag{3-18}$$

where $\{|+\rangle_L, |-\rangle_L\}$ is an alternative basis for the code space. This feature has a connection with the new quantum exclusive error Z in the following way. The definition of code distance is essentially the same for the quantum case: it is *the minimal number of errors required to map a codeword into another*. What is the distance of the 3-qubit code? Considering naively only $\{|0\rangle_L, |1\rangle_L\}$, it appears as if the distance is 3 since we require three bit-flips to transform a codeword into another. Nevertheless, note that a single Z on any of the qubits maps $|+\rangle_L \leftrightarrow |-\rangle_L$, making it a distance 1 code. As aforementioned, the Z error prevents the 3-qubit code from detecting and correcting single errors as its classical counterpart. The possibility of changing basis and the existence

of a phase-flip error arises from quantum superposition, a feature essential for quantum computation not present in the classical case.

Pauli operators which map codewords into codewords or give a relative phase between subsets of codewords are defined as *logical operators* or *logical errors* depending on the context. Notation wise, one replaces the ordinary single-qubit X and Z for the logical $\bar{X}$ and $\bar{Z}$ operators called logical bit-flip and phase-flip, respectively. Logical operators serve as high-level gates to perform quantum computations in the level of the logical states. For the 3-qubit code on the computational basis, e.g., a set of logical operators is

$$\bar{X} = X_1X_2X_3,\ \bar{Z} = Z_1. \tag{3-19}$$

An interesting fact is that the above classification is base-dependent. Consider the logical codewords on the diagonal basis. For this base, Z_1 is a bit-flip and $X_1X_2X_3$ is a phase-flip, as is verified by Eqs. (3-17) and (3-18). Hence, on the diagonal basis:

$$\bar{X} = Z_1,\ \bar{Z} = X_1X_2X_3. \tag{3-20}$$

This is an interesting duality between the two bases where a logical bit-flip (phase-flip) becomes a logical phase-flip (bit-flip) when one expresses the state on the other base.

Finally, for the problem of quantum collapse (3), *stabilizer measurements* are cleverly used to not disturb the logical state. Consider again the 3-qubit code on the computational basis. As aforementioned, a state $|\psi\rangle$ is encoded into $|\psi\rangle_L \in \mathcal{C}_0 = \text{span}\{|000\rangle, |111\rangle\}$ (the subscript will become clear shortly). Note that this subspace is stabilized by $\mathcal{P}_0 = \{Z_1Z_2, Z_2Z_3\}$, i.e., any state in $\mathcal{C}_0$ is eigenstate of both elements of $\mathcal{P}_0$ with eigenvalue $+1$. Consider the logical state after it is afflicted by a bit-flip on one of its qubits. There are three possibilities:

$$\begin{aligned} |\psi\rangle_L &\xrightarrow{X_1|\psi\rangle_L} \alpha|100\rangle + \beta|011\rangle \in \mathcal{C}_1 = \text{span}\{|100\rangle, |011\rangle\} \\ &\xrightarrow{X_2|\psi\rangle_L} \alpha|010\rangle + \beta|101\rangle \in \mathcal{C}_2 = \text{span}\{|010\rangle, |101\rangle\} \\ &\xrightarrow{X_3|\psi\rangle_L} \alpha|001\rangle + \beta|110\rangle \in \mathcal{C}_3 = \text{span}\{|001\rangle, |110\rangle\}. \end{aligned} \tag{3-21}$$

The resulting state pertains to a orthogonal subspace to $\mathcal{C}_0$. This leads us to divide the Hilbert space in four mutually orthogonal subspaces $\mathcal{C}_i$, with $i = 0, 1, 2, 3$. Now, the crucial point is that states pertaining to each $\mathcal{C}_i$ are

stabilized by different sets $\mathcal{P}_i$:

$$\begin{aligned}\mathcal{P}_0 &= \{+Z_1Z_2, +Z_2Z_3\}\\ \mathcal{P}_1 &= \{-Z_1Z_2, +Z_2Z_3\}\\ \mathcal{P}_2 &= \{-Z_1Z_2, -Z_2Z_3\}\\ \mathcal{P}_3 &= \{+Z_1Z_2, -Z_2Z_3\},\end{aligned} \tag{3-22}$$

where the subscript relates to the index of the subspace. We exploit this fact to determine which subspace the state lives in, and consequently which X_i error occurred, by introducing the concept of *syndrome extraction*.

The circuit depicted in Figure 3.3 performs both the EC for the 3-qubit code and the syndrome extraction. The E gate is a possible error that occurred after the EC. For this particular example we take $E \in \{I, X_1, X_2, X_3\}$. In the syndrome extraction section of the algorithm, two measurement ancillae, one for each element of $\mathcal{P}_0$, are attached to the system and pass through a measurement procedure as shown in Figure 3.3(b). Ignoring for the moment the second ancilla, the global state pre-measurement (at the red line) is

$$\frac{1}{2}\left(I + Z_1Z_2\right) X_i \left|\psi\right\rangle_L \left|0\right\rangle_{A_1} + \frac{1}{2}\left(I - Z_1Z_2\right) X_i \left|\psi\right\rangle_L \left|1\right\rangle_{A_1}. \tag{3-23}$$

Since $E\left|\psi\right\rangle_L \in \mathcal{C}_i$, it is stabilized by either $\pm Z_1Z_2$. Therefore, the state of the ancilla qubit is *deterministically* $\left|0\right\rangle_{A_1}$ or $\left|1\right\rangle_{A_1}$ before measurement. Hence, by measuring the state of the first ancilla, one learns the phase of the stabilizer without disturbing the encoded state, effectively performing a non-demolition measurement. The same reasoning goes for the second ancilla, where one learns the phase of the second stabilizer Z_2Z_3. These two pieces of information are sufficient to determine which subspace $E\left|\psi\right\rangle_L$ lives in leading to a proper correction prescription.

3.2 Stabilizer Codes

The 3-qubit code is an interesting example despite not being a practical QECC. It encapsulates all the adaptations needed to circumvent problems brought up by the superposition principle, wavefunction collapse, and the no-cloning theorem. It is almost a suitable QECC since it can detect and correct single bit-flips, making it actually a proper classical code, but the new quantum exclusive phase-flip error renders it incomplete.

In this section, we generalize the underlying procedure behind the 3-qubit code, which is an example of a *stabilizer code*, an important class of QECCs introduced by Gottesman [42, 61]. In section 3.2.2, we show Shor's code [41],

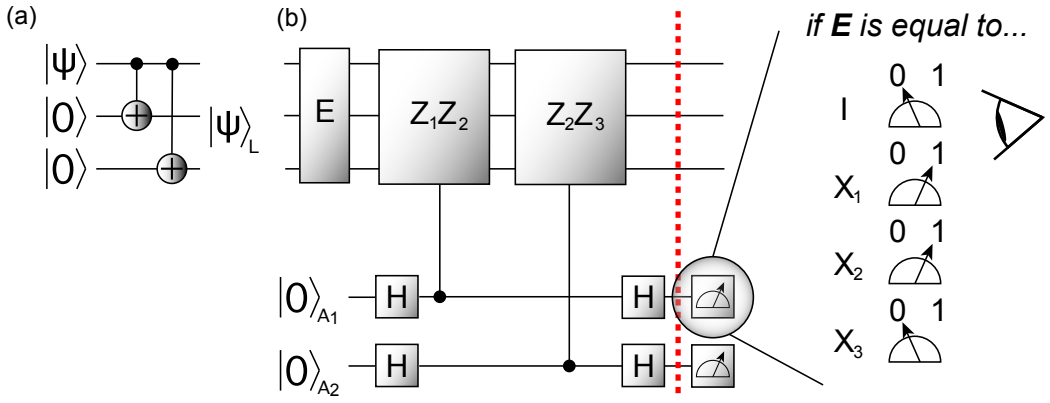

Figure 3.3: 3-qubit code encompassing the encoding and the syndrome extraction stages. (a) Encoding stage; (b) Syndrome extraction stage where two ancillae qubits are attached to the main block to measure the syndrome of a possible error.

the most famous example of a QECC. In section 3.2.3, we prove an inequality relating the Hilbert space dimension of the code and the minimum qubit overhead for non-degenerate codes correcting up to t errors. Subsequently, the perfect code is briefly introduced.

3.2.1 General theory of stabilizer codes

The main objective of a QECC is to protect k data-qubit registers from a set of errors $\mathcal{E}$. By protection, it is meant that the code must be able to detect and correct any error in $\mathcal{E}$. Figure 3.4 summarizes the general structure of a $[[n, k, d]]$ stabilizer error correction code. It is divided into three stages: *encoding*, *syndrome extraction* and *correction*. We now describe each stage in detail.

On the encoding stage, Figure 3.4(a), a k-qubit data-state $|\psi\rangle_D$ is entangled with $m = n - k$ auxiliary qubits via an EC forming a n-qubit logical state $|\psi\rangle_L$. The EC defines the set of codewords $\mathcal{C} = \{|c_i\rangle_L\}_i$ which specify how the other stages of the code function. In general, $|\mathcal{C}| = 2^k$. For example, for $k = 2$, the encoding stage maps $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$ into four mutually orthogonal codewords in an expanded Hilbert space.

During syndrome extraction, errors are detected by performing m syndrome measurements as shown in 3.4(b). The codewords defines a group of common stabilizers spanned by m generators [36]. Let $\mathcal{P} = \{P_i\}$ be a given set of generators for the common stabilizer group for $\mathcal{C}$. We denote the elements of $\mathcal{P}$ syndrome operators. It follows that syndrome operators satisfy the following properties [39]:

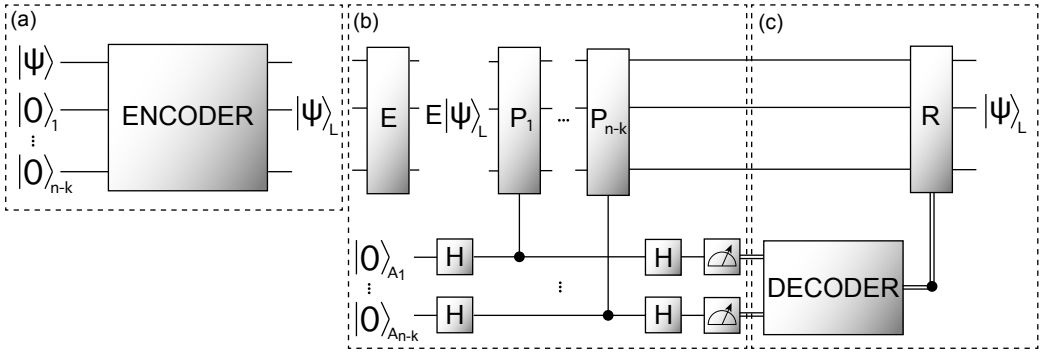

Figure 3.4: The generic circuit of a $[[n, k, d]]$ stabilizer error correction code. (a) A data register $|\psi\rangle_D$ is entangled with $n-k$ redundancy qubits via an EC to form the logical-state $|\psi\rangle_L$. (b) After a potential error E occurs, ancilla qubits are attached to $|\psi\rangle_L$ and m syndrome measurements P_i are performed. The result of the measurements produces the syndrome. (c) With the syndrome, one consults the syndrome table, and the appropriate correction R is appointed and applied. This process is represented by the decoder gate. The double-line channels means classical communication.

1. $\mathcal{P} \subseteq \mathcal{G}_n$;
2. $P_i |\psi\rangle_{L,j} = +1 |\psi\rangle_{L,j} \; \forall \, i, j$;
3. $[P_i, P_j] = 0 \, \forall \, i, j$,

where $\mathcal{G}_n$ is the general n-qubit Pauli group. Property 2 ensures that the syndrome measurements do not further disturb the damaged logical state, and property 3 allows one to perform measurements in any order.

Let $E \in \mathcal{E}$ be an error that occurred between the encoding and syndrome extraction stages. The effect of each syndrome measurement is to map $E |\psi\rangle_L$ into the superposition

$$E |\psi\rangle_L |0\rangle_{A_i} \rightarrow \frac{1}{2} \left[(I + P_i) E |\psi\rangle_L |0\rangle_{A_i} + (I - P_i) E |\psi\rangle_L |1\rangle_{A_i} \right]. \tag{3-24}$$

Note that E necessarily either commutes or anti-commutes with S_i since $E, P_i \in \mathcal{G}_n$ (see property 4 on Appendix C). If E and P_i commutes (anti-commutes), the final state is unequivocally $E |\psi\rangle_L |0\rangle_{A_i}$ ($E |\psi\rangle_L |1\rangle_{A_i}$). Therefore, each syndrome measurement can be understood as a deterministic measurement of the state with the outcome reveling whether the error commutes or anti-commutes with the syndrome operator. At the end of the syndrome extraction stage, one is left with a binary syndrome string of length m whose i-th entry encodes whether P_i and E commutes or not.

Table 3.1: 3-qubit code syndrome table for single-qubit errors.

Error	Syndrome	Error	Syndrome
X_1	10	Z_1	00
X_2	11	Z_2	00
X_3	01	Z_3	00

Given $\mathcal{E}$ and $\mathcal{P}$, one builds the so-called *syndrome table* relating each error to the corresponding syndrome string it generates. As an illustration, Table 3.1 shows the syndrome table for the 3-qubit code with $\mathcal{P} = \{Z_1Z_2, Z_2Z_3\}$, and considering errors with weight one on three qubits. Note that the syndromes for Z_i are composed only of zeros, which is the same syndrome for $E = I$ since the identity commutes with any operator. These errors are classified as *undetectable* [36] as they are not distinguishable from I.

Finally, in the correction stage one prescribes an operator R for which

$$RE\left|\psi\right\rangle_L = \left|\psi\right\rangle_L . \tag{3-25}$$

Since Pauli operators square to the identity, R is in principle identical to the appointed error guided by the syndrome table. The decoder gate of Figure 3.4(c) is a crosscheck, performed in a classical computer, between the extracted syndrome and its related error on the syndrome table. This scheme functions perfectly for *non-degenerate codes*, where a one-to-one correspondence between errors and syndromes exists. On the other hand, for *degenerate* codes multiple errors can produce the same syndrome. For correction to be successful, all two-on-two combinations of errors with the same syndrome must stabilize all codewords. Consider an arbitrary correction code with codewords $\{\left|c_i\right\rangle_L\}$, and let $\{E_i\}$ be a set of errors with the same syndrome. One requires that

$$E_iE_j\left|c_k\right\rangle_L = +\left|c_k\right\rangle_L \ \forall i,j,k \tag{3-26}$$

for $\{E_i\}$ to be correctable. If the above condition is met, applying any element of $\{E_i\}$ will restore the logical state even though it is impossible to single out which error actually took place. If for some pairing E_iE_j Equation (3-26) is not satisfied, the set $\{E_i\}$ is classified as *uncorrectable*, since it is impossible to decide the proper correction operation.

3.2.2 Shor's code

We now demonstrate the concepts presented in the previous section with a fully functional code, the celebrated Shor's code devised by Peter Shor [41]. Classification wise, it is a $[[9,1,3]]$ code encoding 1 data qubit into 9 logical

qubits on a distance 3 code space. Figure 3.5 shows an EC for the QECC, which maps the computational basis into the codewords

$$|0\rangle_L = \frac{(|000\rangle_{123} + |111\rangle_{123})}{\sqrt{2}} \frac{(|000\rangle_{456} + |111\rangle_{456})}{\sqrt{2}} \frac{(|000\rangle_{789} + |111\rangle_{789})}{\sqrt{2}}. \quad (3\text{-}27)$$

$$|1\rangle_L = \frac{(|000\rangle_{123} - |111\rangle_{123})}{\sqrt{2}} \frac{(|000\rangle_{456} - |111\rangle_{456})}{\sqrt{2}} \frac{(|000\rangle_{789} - |111\rangle_{789})}{\sqrt{2}}. \quad (3\text{-}28)$$

Note that the codewords are products of GHZ states [60]. A possible set of syndrome operators is

$$\begin{aligned}\mathcal{P} = \{&Z_1Z_2, Z_2Z_3, Z_4Z_5, Z_5Z_6, Z_7Z_8, Z_8Z_9, \\ &X_1X_2X_3X_4X_5X_6, X_4X_5X_6X_7X_8X_9\}.\end{aligned} \quad (3\text{-}29)$$

Since $\mathcal{P}$ can be written as a set whose elements contain only X or Z, Shor's code belongs to the CSS class [62] of QECCs named after its inventors Robert Calderbank, Peter Shor and Andrew Steane. CSS codes play an important role in quantum error correction since the problem of decoding is simplified for them [62, 63], and many important topological surface codes pertain to the CSS class [43, 64].

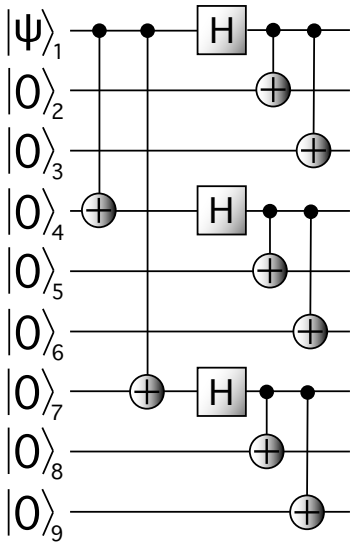

Figure 3.5: EC for Shor's quantum error correction code.

From $\mathcal{P}$, and only considering single-qubits errors, we build the syndrome table shown in Table 3.2. Note that all X_i errors are perfectly distinguishable as each one generates a different syndrome. On the other hand, there are only three distinct syndromes for the Z_i errors, which makes Shor's code a *degenerate* QECC. We can make a supplementary table as shown in Table 3.3 to group degenerate syndromes with their common errors. Although the

Table 3.2: Shor's code syndrome table for single-qubit errors.

Error	Syndrome	Error	Syndrome
X_1	10000000	Z_1	00000010
X_2	11000000	Z_2	00000010
X_3	01000000	Z_3	00000010
X_4	00100000	Z_4	00000011
X_5	00110000	Z_5	00000011
X_6	00010000	Z_6	00000011
X_7	00001000	Z_7	00000001
X_8	00001100	Z_8	00000001
X_9	00000100	Z_9	00000001

Table 3.3: Auxiliary table grouping degenerate syndromes with respectively associated errors. For each syndrome, all pairing combinations of errors belongs to the common stabilizer group of the codeword.

Syndrome	Errors
00000010	Z_1, Z_2, Z_3
00000011	Z_4, Z_5, Z_6
00000001	Z_7, Z_8, Z_9

syndromes for Z errors do not allow us to point out which error occurred, correction is possible because all Z_iZ_j combinations belong to the group of mutual stabilizers to the codewords.

3.2.3 Qubit overhead and the Perfect code

Before presenting the perfect code, it is interesting to prove the minimum qubit overhead a QECC must have to correct up to t errors to understand why the code is considered perfect. The proof is for non-degenerate codes.

Let n be the Hilbert space dimension of the code. Considering errors as a product of Pauli operators, we express a general error by assigning a Pauli letter for each entry of a vector of the form $\boldsymbol{E} = (a_1, a_2, \ldots, a_n)$. Define $n_e(n,t)$ the number of errors with weight t. For $t = 1$, each possible error can be constructed by choosing one of the n entries of $\boldsymbol{E}$ and assigning one of three Pauli letters $\{X, Y, Z\}$. Therefore,

$$n_e(n,1) = \binom{n}{1} \times 3. \tag{3-30}$$

For $t = 2$ the reasoning is similar: we choose two entries in n to allocate the errors and, for each entry, we choose one of three Pauli letters. Thus,

$$n_e(n,2) = \binom{n}{2} \times 3^2. \tag{3-31}$$

This reasoning holds true for any $t \leq n$. Therefore, the general formula for $n_e(n,t)$ is given by

$$n_e(n,t) = \binom{n}{t} \times 3^t. \tag{3-32}$$

For error-correction we are interested in detecting and correction up to t errors. Let $s(n,t)$ denote all possible errors up to t errors, i.e., the number of errors with weight less or equal to t. It follows that

$$s(n,t) = \sum_{i=1}^{t} n_e(n,i) = \sum_{i=1}^{t} \binom{n}{i} \times 3^i. \tag{3-33}$$

With $s(n,t)$ we can derive the minimum number of qubits necessary for constructing a non-degenerate quantum error-correction code capable of handling errors up to a weight t.

The argument goes as follows: if the codewords are made by n qubits, then at most $n-1$ are auxiliary ancilla qubits employed in the syndrome measurement stage. Therefore, the syndrome is a vector with at most $n-1$ binary entries. Since there exist 2^{n-1} binary vectors with $n-1$ entries and at least one distinct vector must be assigned to each particular error, there must exist at least as many binary vectors as the number of possible errors for a non-degenerate error correction code to be able to correct all errors up to weight t:

$$s(n,t) + 1 \leq 2^{n-1} \tag{3-34}$$

The 1 added on the LHS is to account for the case where no errors occurred. In particular, for $t = 1$ it follows

$$\binom{n}{1} \times 3 + 1 = 3n + 1 \leq 2^{n-1}. \tag{3-35}$$

Note that $n = 5$ saturates the inequality (3-35), therefore it is of no use to try to build a QECC with less then 5 qubits. Non-degenerate 5-qubits codes that correct single-qubit errors are called perfect codes [36, 40] since they have the property of using every available syndrome for 5 qubits. As an example, Table 3.4 shows the syndrome table of a perfect code that was evolved by the GA that will be presented in the next chapter. The Y_i errors were added to the table to make it clear the use of all syndromes available. It is common to omit the Y_i errors since they are the combination of X_i and Z_i (note that the syndromes for Y_i is the bitwise modulo 2 sum of the syndromes for X_i and Z_i).

Table 3.4: Example of syndrome table of a perfect code.

Error	Syndrome	Error	Syndrome	Error	Syndrome
X_1	1101	Y_1	0110	Z_1	1011
X_2	1000	Y_2	0001	Z_2	1001
X_3	0010	Y_3	0111	Z_3	0101
X_4	0011	Y_4	1111	Z_4	1100
X_5	1110	Y_5	0100	Z_5	1010

4
Genetic algorithms applied to Quantum Circuits

For about 4.28 billion years [65], millions of living species go through a continuous optimization process, what Darwin termed the evolution of species [66]. The evolutionary process occurs through the interaction of populations of species with the habitat they live whose function is to select individuals with a greater aptitude to grow, thrive, and reproduce. From a mathematician perspective, one can allude that the environment determines a fitness function whose domain and codomain are individuals of a population and a measurement of how well an individual is adapted to the environment, respectively. The fitness function then becomes the cost function of the natural selection optimization problem.

A key attractive point of evolution is its robustness to noise, a feature that is often missing in standard optimization methods [67]. The environment is not static: through natural unpredictable events, it transforms in a chaotic fashion leading to an ever-changing fitness landscape. Hence, inspired by the evident success of life in thriving through evolution, researchers have been using algorithms simulating natural selection to solve mathematical optimization problems as early as 1957 [68–74] broadly termed evolutionary algorithms [67]. Mimicking natural evolution is nothing new to humanity. It is estimated that artificial selection processes, such as selective breeding, have been conducted by humans since prehistoric times [76–78], making it natural to extend its usage to computer programs. Evolutionary algorithms is an umbrella term for several types of algorithms that, in some way, mimic natural evolution [75]. We work mainly with GAs.

This chapter is organized as follows. In Section 4.1, we make a brief introduction to the biological evolutionary process emphasizing its genetic point of view. Following, GAs are introduced in Section 4.2 adapted to evolve quantum circuits, in particular Clifford circuits. The chapter culminates with the application of GAs to evolve QECCs in Section 4.3, which is this work's central topic of interest.

4.1 The evolutionary process

Given a population of individuals belonging to the same species, one may divide its evolution process into four stages [67]: reproduction, mutation, competition, and selection. In broad terms, reproduction is the transmission of genetic material from parents to their offspring. Mutations are minute stochastic errors that occur during the transmission of genes. Competition and selection drift the population towards individuals better adapted to the environment, where the fittest individuals reproduce passing along their genes to future generations, while the unfitted perish removing their genes from the gene pool. An evolutionary algorithm aims at emulating, to some extent, this cycle for a population of potential solutions to an optimization problem until a threshold is reached.

The evolution process as defined above hinges on the representation of individuals as their genetic material, commonly referred to as the *genotype*, and a screening method that favors certain genotypes over others. An essential point is that the genotype is a collection of elementary building blocks from a finite set of possibilities. For living beings, the genotype is a set of DNA strands, a polymer made of polynucleotide chains. Polynucleotide chains are arbitrary sequences of four nucleotides types: adenine (A), thymine (T), guanine (G), and cytosine (C) [79]. Hence, a DNA strand may be represented by its nucleotide sequence as, for example, "ATTGCGA...". Considering sexual reproduction (reproduction involving two mates where the offspring's genetic material is a mix of the parent's genotypes [80]), the genotype design naturally leads to the *genetic operations* of *crossover* and *mutation*. Crossover is a recombination of the parent's DNA, where whole segments of DNA are interpolated to form the offspring's DNA at conception. Crossover promotes the dispersion of good DNA blocks among the population, as natural selection gradually removes bad blocks, and mutation enables the search of new solutions not reachable through recombination. Figure 4.1 shows simple examples of a one-point crossover between two DNA strands and mutation of single genes.

Given the genetic operators, we have only an aimless, random walk through the genotype sample space due to their stochastic nature. Natural selection is responsible for drifting the walk in the direction of genotypes that beget fitter individuals. Natural selection results from the clash between the environment's restrictions on the individuals and the *phenotype* each one displays. The phenotype is the genetic expression of the genotype, i.e., the set of physical attributes displayed by the individual [81]. For example, in a habitat with high trees, giraffes with genes that generates long necks have a competitive

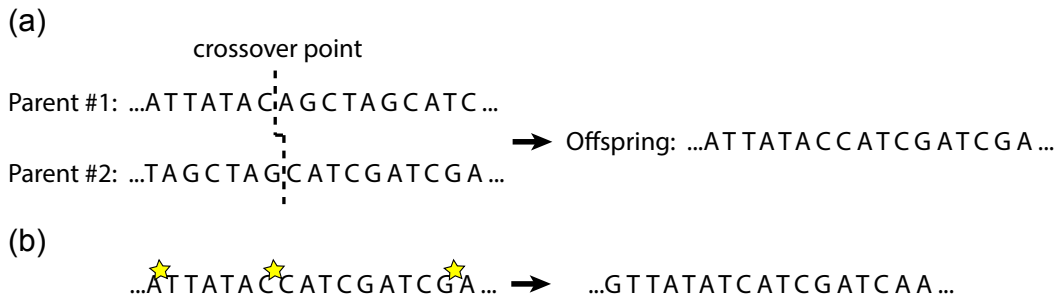

Figure 4.1: Genetic operations examples. (a) a one-point crossover between two DNA strands. The strands are cut at the crossover point and combined together to form the offspring's DNA. (b) three random mutations, symbolized by the yellow stars, on a DNA strand. Due a copy error, some nucleotides are replaced by random ones.

advantage over giraffes with short necks, thus having a higher probability of mating. After several generations, the genotype set of the population will drift to have genes that generate giraffes with necks proper to the size of the trees, optimizing the individuals to the environment.

A helpful way of visualizing the evolutionary optimization process is with a fitness landscape [82, 83]. The fitness landscape is a mathematical construct that maps genotypes to reproduction rates. The environment defines a function in which the domain is the space of all possible genotypes and whose codomain is the reproduction rate. Distances on the landscape are defined as the closeness of two genotypes, i.e., how similar are the phenotypes they spawn. Figure 4.2 exemplifies a representation of an arbitrary fitness landscape. The rationale is that, at an initial time, the population genotype set is centered at some point in the landscape. As the generations go by, the population stochastically drifts towards the surface peaks due to natural selection and the genetic operators. Naturally, the fitness landscape is a metaphor, as actually producing such a graph is impossible due to the complexity of the problem.

4.2 Genetic algorithms applied to Clifford circuits

We now describe how we mimic nature's evolutionary process in a GA applied to quantum circuits. Since the access to fully-fledged quantum computers is not a reality yet, we work mainly with Clifford circuits which we can efficiently simulate with classical computers.

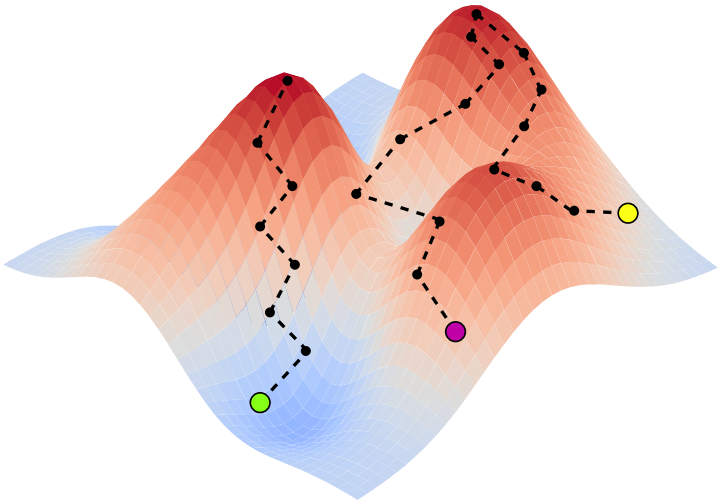

Figure 4.2: Illustration of three distinct populations climbing a hypothetical fitness landscape. At an initial time, each population starts at some low point of the landscape (the colored circles) and stochastically rises towards the peaks. The mutation rate regulates the size of each step. The height of the surface represents the reproduction rate of a given genotype.

4.2.1 The algorithm

In light of what was discussed in the previous section, we acknowledge four main aspects we need to cover to build a GA that simulates nature:

- a genetic representation of the tentative solutions, which we also call the genotype of the individual;
- a method to implement the genetic operators of crossover and mutation;
- a selection mechanism that distinguishes between solutions regarding the optimization problem;
- a competition mechanism to remove inferior genotypes from the gene pool.

It turns out that it is straightforward to design a genetic representation for quantum circuits. To illustrate this point and showcase how we genetically represent quantum circuits in this work, consider the random circuit depicted in Figure 4.3(a) and its genetic expression in Figure 4.3(b). We genetically represent a circuit composed of t gates as a $t \times 3$ array whose rows are gates in ascending order of application. The first column stores the operator, and the second and third the indices of the affected qubits. If the gate is a CNOT, the second (third) column is the control (target) qubit index. For single-qubit gates the third column is ignored.

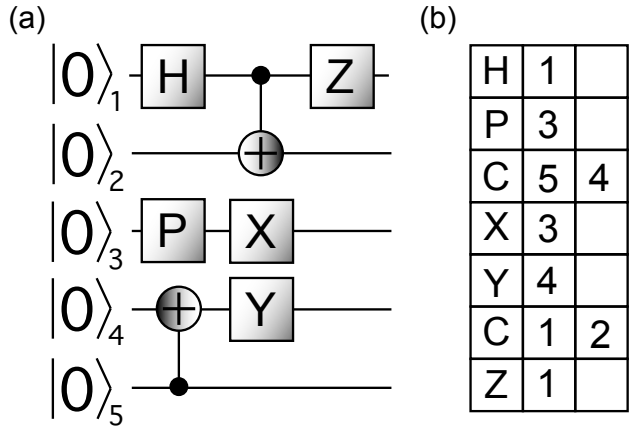

Figure 4.3: A (a) random quantum circuit and (b) its genotype. Each row of the genotype is a gate in ascending order (from top to bottom) of application with the columns storing the operator and the indices of the affected qubits. The CNOT is abbreviated as C.

Crossovers and mutations naturally become row operations by expressing quantum circuits as a matrix. There is a freedom of choice to define how each genetic operator works. The most appropriate option often arrives from experimentation since the efficiency of a GA is strongly dependent on the optimization problem itself [75]. For instance, the crossover can be performed at one-point (as illustrated in Figure 4.1(a) for a DNA strand) or at multiple points [67, 75, 84]. Additionally, we may use more than two parents at the reproduction step, thus having more than two genotypes to crossover [85, 86].

To simplify, we worked solely with one-point crossovers between the genotypes of two parents. Given two arbitrary genotypes, we perform a crossover by conducting the following steps:

1. a split point is randomly chosen for each parent with a uniform probability distribution;

2. ordering the parents as A and B, their genotypes are divided at the split point. Offspring A(B) is formed by stacking the top portion of parent A(B) on top of the bottom portion of parent B(A).

Figure 4.4 shows a crossover example. Note that two offspring are always generated.

Mutations happen to the offspring's genotype after the crossover procedure. We define single mutations as follows:

1. with equal probability, it is chosen (a) if the mutation will modify an existing gate or (b) if it will insert a new gate into the circuit;

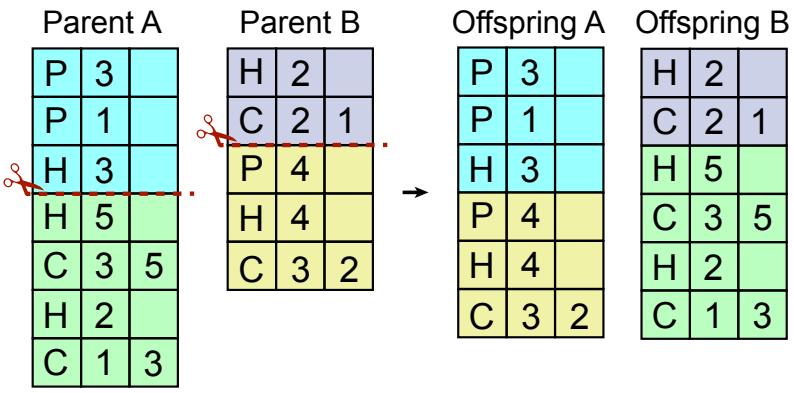

Figure 4.4: Example of a crossover between two genotypes. The parents genotypes are divided at randomly chosen points and their offspring are built by stacking the pieces.

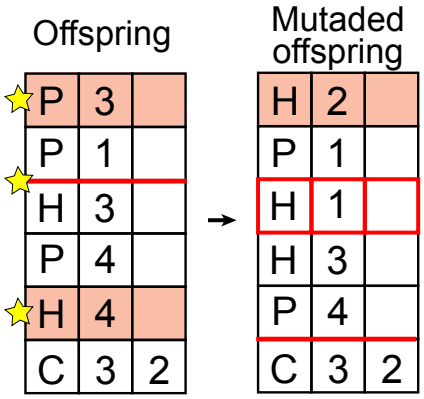

Figure 4.5: Illustration of the three types of mutation that may occur to a circuit. From top to bottom: the first gate was replaced by a Hadamard on qubit 2, a new row was inserted between the second and the third rows, and an identity replaced the fifth gate (hence it was deleted).

(a) if (a), a random row of the genotype is uniformly selected to be altered. The row contents are overwritten by a new gate uniformly selected between $\{I, H, P, \text{CNOT}\}$. If the identity is picked, the entire row is deleted;

(b) if (b), a random insertion point on the genotype is uniformly selected. A new row with a new uniformly selected gate chosen between $\{H, P, \text{CNOT}\}$ is inserted at the insertion point.

Figure 4.5 displays an example with the three kinds of mutations that can happen to an offspring genotype.

Just as in nature, we also consider that each genotype gives rise to a phenotype in GAs. We regard the phenotype of a given circuit as a set of real-valued numbers that quantify its properties. For instance, we extensively consider the depth (the length of the longest path from the beginning of a

circuit to its end) as an essential parameter since it is a measure of how long it takes to execute a circuit. Thus, from a population of circuits, one can sort the individuals by their depth and use it as a selection bias, e.g., considering circuits with lower depth as better suited. Hence, the *selection mechanism* for quantum circuits works in the following way:

1. The phenotype of each individual is evaluated, where the optimization problem determines the list of parameters assessed;

2. Each individual's fitness is evaluated via a fitness function $\mathcal{F}$ whose arguments are the elements of the phenotype. By convention, we define $\mathcal{F}$ such that higher fitness individuals are considered better solutions;

3. For breeding selection, a probability of reproduction is associated to each individual proportional to its relative fitness to the rest of the population. A roulette wheel selection system [67] is employed to pick two individuals to mate, i.e., to go through the process of crossover/mutation.

The most computationally expensive part of the GA is calculating the phenotype. Also, it can be a creative challenge to come up with a proper list of parameters. One needs to clearly understand the optimization problem to devise a list of desirable parameters the optimal solution should have and a method to evaluate them quantitatively.

Finally, after the size of the population reaches an established maximum limit (reproduction adds two new individuals at a time), we may purge the worst circuits. We regard this process as the competition aspect of the GA since it emulates the limited growth of biological populations and the death of ill-adapted individuals.

With all the fundamental elements of the GA defined, we build the basic scheme of the simulated evolutionary cycle. At the start, an initial population of random circuits is initialized with each individual's fitness evaluated and stored. The GA then works via the iteration of a cycle of selective breeding, crossover, mutation, and population purge until a termination criterion is met[1]. Each cycle marks a *generation*. Putting it all together, Figure 4.6 displays the decision tree of the GA we built.

[1]The criterion can be a maximum number of iterations or a target fitness value.

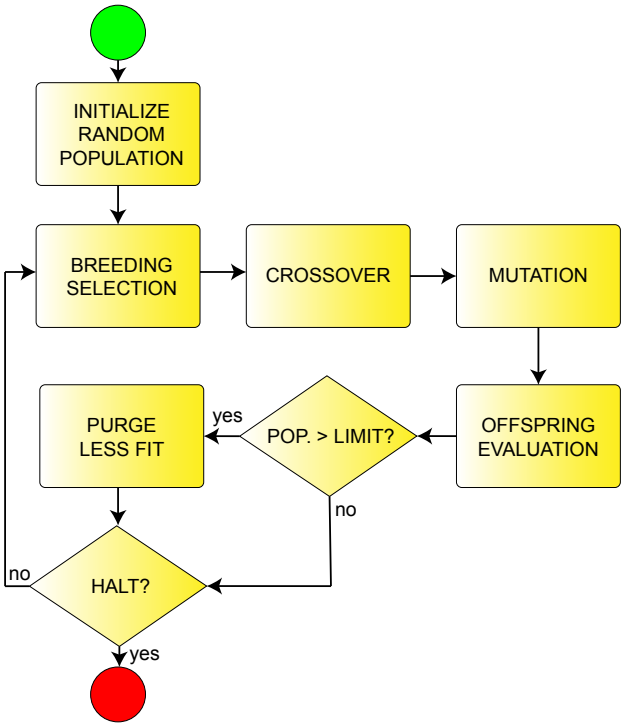

Figure 4.6: Decision tree of the genetic algorithm. The halt decision gate evaluates if a termination condition was met.

4.2.2 Dynamic crossover and mutation rates

In addition to the essential elements of the GA, one usually adds a series of heuristics to increase the algorithm's performance, i.e., how quickly it finds a solution [67,75]. Unlike the usual optimization methods where the algorithm has a certain independence from the problem itself, GAs are very problem sensitive, requiring a degree of trial and error since there is a considerable degree of arbitrariness on how each step is carried out. As aforementioned, e.g., there a multiple ways to perform the crossover operator; we may choose between one-point or multiple-point crossover, or crossover genotypes of more than two parents. One is not restricted to just these choices for new forms of crossover can be invented.

Fogel [67] stresses that a diverse gene pool is essential. As generations go by, the population diversity diminishes due to selective breeding in conjunction with crossover. Crossovers between the best individuals have a homogenization effect since it keeps recombining increasingly similar genotypes until the population is composed of identical individuals (considering the worst genotypes are periodically deleted). There are some proposed ways to support a high diversity, such as working with large-sized populations, introducing new random individuals periodically, or avoiding breeding similar genotypes [67].

On the other hand, the evolutionary algorithm termed *evolution strategies* (ES) contests the need for high diversity [75, 87]. In its basic form, ES algorithms operate with only two individuals at a given time: a parent genotype and a mutated copy of it. Since crossovers become pointless, ES works by constantly copying and mutating a single genotype until it generates an individual with higher fitness, which replaces its parent. We regard the ES approach as similar to simple organisms performing asexual reproduction [80], like bacteria on a Petri dish.

We investigated some of the proposed heuristics to increase diversity, but we did not observe increased performance by working with a large population or introducing new random circuits periodically. New random individuals are seldom chosen to breed since they usually have low fitness, and the diversity a large population brings is eventually washed out by selective breeding and crossover. We did not venture into avoiding breeding similar genotypes since it would demand the extra step of computing a measure of distance[2], which can be computationally expensive. We settled in using a low-sized population (≈ 10) with dynamic crossover and mutation rates.

Ulteriorly, the genetic operators are scanning mechanisms to survey the solution space. Bartz-Beielstein et al. [75] concisely explain the difference in purpose of each operator:

> Mutation means neighborhood based movement in the search space that includes the exploration of the 'outer space' currently not covered by a population, whereas recombination rearranges existing information and thus focuses on the 'inner space'.

Given the distinction, we argue that the need for crossovers and mutations varies as the algorithm progresses. Since crossovers homogenize the population by disseminating good blocks of genes found on the 'inner space', they are most relevant when diversity is high. As diversity drops, mutations become increasingly important (crossovers start to have no effect since they merely copy the parents' genotypes) as new gene combinations need to be sought out on the 'outer space'. Hence, for a given population of circuits, define the fitness register $\vec{f}_{\text{reg}}(t)$ as an array that holds each individual's fitness at a given time t. Let $f_{\text{s.d.}}(t)$ be the standard deviation of $\vec{f}_{\text{reg}}(t)$. We use $f_{\text{s.d.}}$ as a rough measure of the population diversity. Define the dynamic crossover (c_r) and mutation (m_r) rates which give the probability that the respective genetic

[2]For Clifford circuits we could use the inner-product as described in Algorithm 2.

operator happens at each reproduction instance:

$$c_r = 1 - e^{-f_{\text{s.d.}}} \tag{4-1}$$

$$m_r = e^{-f_{\text{s.d.}}} \tag{4-2}$$

Thus, the algorithm adapts at each iteration, providing the necessary genetic operator more importance. Additionally, we added stochasticity to the number of mutations an offspring undergoes. Each time mutations are to be applied, the quantity is drawn from a probability distribution.

Given the discussed heuristics and the decision tree of the GA, Algorithm 5 presents the pseudocode of the GA we employ in this work.

4.2.3 A toy problem

As a first test of the capability of the GA as a searching tool for quantum circuits we introduce a simple toy problem. A circuit U acts on a 1D lattice of qubits, which we now label by the indices $x \in \{1, ..., n\}$, initially in the state $|\psi_0\rangle = |0\rangle^{\otimes n}$. The resulting state after application of the circuit is $|\psi\rangle = U|\psi_0\rangle$. Define the density matrix ρ_x as the state obtained by splitting the chain at x and tracing out all qubits to the left of x in the final state $|\psi\rangle$. The von Neumann entropy of ρ_x is denoted $S(x)$, and we define the *mean entropy* generated by a circuit as

$$\langle S\rangle = \frac{1}{n}\sum_x S(x). \tag{4-3}$$

For Clifford circuits, Algorithm 4 can be employed to evaluate $\langle S\rangle$.

Consider then the following problem: *find quantum circuits that generate the largest possible $\langle S\rangle$ with the least possible circuit depth D.* We propose the fitness function

$$\mathcal{F} = \langle S\rangle / D \ . \tag{4-4}$$

Circuits that maximize (4-4) solve the problem. Note that solutions can be found by deductive reasoning as follows. Subadditivity of the von Neumann entropy implies that $S(x)$ can change by at most one from one qubit to the next [56],

$$|S(x+1) - S(x)| \leq 1. \tag{4-5}$$

The maximum mean entropy of a circuit is therefore given by

$$\langle S\rangle_{\max} = \frac{1}{n}\left(\sum_{M=1}^{n/2} M + \sum_{M=1}^{n/2-1} M\right) = \frac{n}{4}. \tag{4-6}$$

The minimum value of depth for a circuit generating non-vanishing entropy is $D_{\min} = 2$. Hence, the highest possible value for $\mathcal{F}$ is $\mathcal{F}_{\max} = n/8$. There are multiple solutions that maximize $\mathcal{F}$, one example of which is shown in Figure

Algorithm 5 Genetic Algorithm

Input: (i) initial population size M; (ii) qubit overhead n; (iii) mutation probability density mut_prob, (iv) population maximum size max_size
Output: circuit with highest fitness

```
1: population = initializePopulation(M, n)
2: while terminationTest() == False do
3:     f_reg = evaluatePopulationFitness(population)
4:     f_s.d. = standardDeviation(f̃_reg)
5:     c_r = 1 − e^{−f_s.d.}
6:     m_r = e^{−f_s.d.}
7:     parent_A, parent_B = selectParents(population, f̃_reg)
8:     if crossoverTest(c_r) == True then
9:         offspring_A, offspring_B = crossover(parent_A, parent_B)
10:    else
11:        offspring_A = parent_A
12:        offspring_B = parent_B
13:    end if
14:    if mutationTest(m_r) == True then
15:        n = mutationNumber(mut_prob)
16:        for i ∈ {1, ..., n} do
17:            offspring_A = mutate(offspring_A)
18:        end for
19:    end if
20:    if mutationTest(m_r) == True then
21:        n = mutationNumber(mut_prob)
22:        for i ∈ {1, ..., n} do
23:            offspring_B = mutate(offspring_B)
24:        end for
25:    end if
26:    population = population + offspring_A + offspring_B
27:    if populationSize(population) > max_size then
28:        population = purgeWorstCircuits(population)
29:    end if
30: end while
```

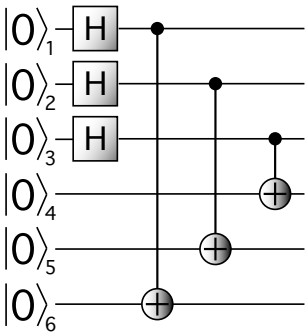

Figure 4.7: A 6-qubit circuit solving the toy problem. Permutations of the CNOTs along the time direction, as well as of the target qubits yield the same solution.

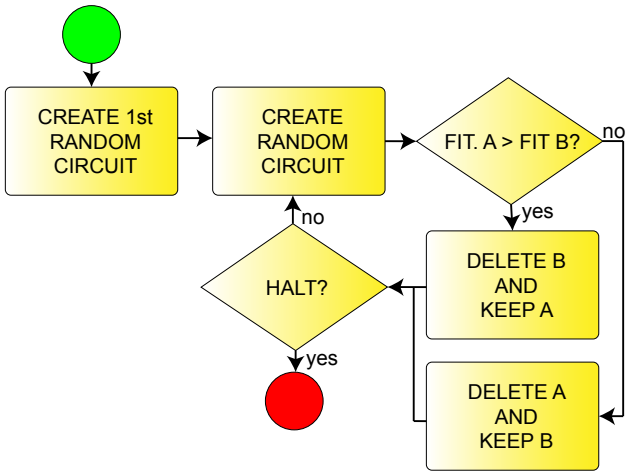

Figure 4.8: Decision tree of the RS. The algorithm generates random circuits and saves the best circuit ever generated until a termination condition is reached (maximum number of generations or maximum fitness value).

4.7 for $n = 6$ qubits. Note that the circuit produces a tensor product of Bell pairs organized in a specific way within the 1D lattice.

To determine whether a GA provides advantage in the search for the solution, we compare its performance to a purely random search (RS). The decision tree used for RS is shown in Figure 4.8. Since the bottleneck of both algorithms is the fitness evaluation, their time performance is nearly identical, i.e., a generation for each method takes almost the same time to execute. Hence, we chose to compare the methods by how fast each can converge to a solution within a given number of generations.

The total number of solutions to the problem is very small in comparison to all possible n-qubit circuits composed of t gates. We expect further that the ratio of solutions to possible circuits decreases significantly with the number of qubits n. If the GA is to have an advantage over RS, we should expect this edge to grow as we increase n. Figure 4.9 shows the fitness values of the best

circuit in the population as a function of generation number for three cases with increasing number of qubits given by $n = 4, 8$ and 16. We note that each curve is the average over 100 runs, and that over all runs the optimal circuits are generated many times by the GA. For $n = 4$ we can see that RS is able to find the solution. Nevertheless, the GA has an increasing advantage over RS as we increase the number of qubits, as expected. This can be seen by the widening gap between the GA and RS traces in Figures 4.9(a), (b) and (c). The red dashed lines show the maximum attainable fitness values, for reference.

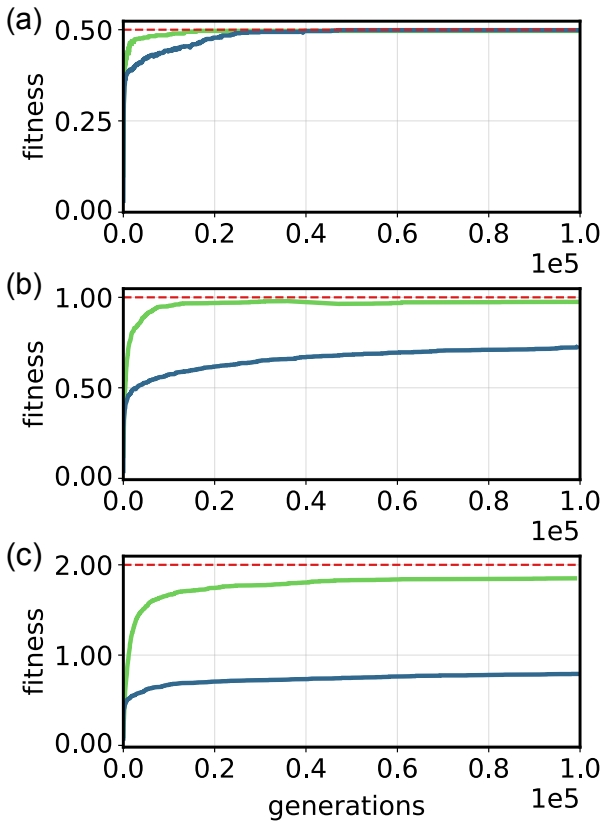

Figure 4.9: Evolutionary search through a genetic algorithm (green line) versus random search (blue line) for the toy problem. Each curve is the average over 100 runs. The dashed red line represents the maximum fitness given by $n/8$ (see Eq. (4-6) and the main text). (a) $n = 4$; (b) $n = 8$; (c); $n = 16$.

We can visualize the topology of a quantum circuit as a graph, where each node corresponds to a qubit and vertices represent CNOT gates. This graph representation highlights the necessary hardware architecture a quantum computer needs to have to execute the circuit, i.e. which qubits need to interact. Figure 4.10 shows the typical example of an initial randomly generated circuit topology, and its subsequent evolution towards the topology of the optimal solution, as depicted in Figure 4.7 corresponding to Bell pairings of the qubits. We see that starting from complicated random circuits artificial selection can

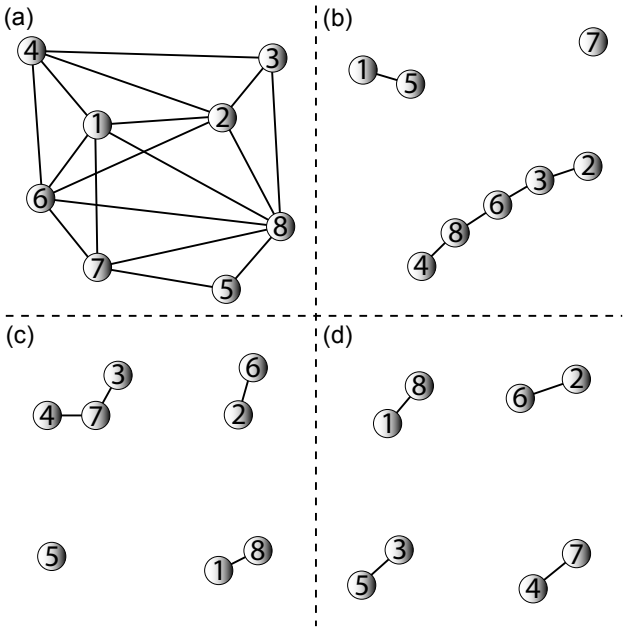

Figure 4.10: Evolution of the topology of the fittest circuit in an evolutionary search simulation for $n = 8$. The edges represent qubits (the number is the qubit's index), and the vertices CNOT gates. (a) Initial set of random circuits; (b) after 322 generations; (c) after 2248 generations; (d) after 3909 generations.

perform a directed search towards high fitness individuals in a much shorter time than RS.

4.3 Evolving QECCs

Given the appropriate fitness function, can a GA evolve quantum error correction codes such as the 5-qubit *perfect* code [40], the 7-qubit *color* code [42, 43], and Shor's 9-qubit code [41]? These are examples of stabilizer codes, meaning they are generated by Clifford gates. As in the toy example, efficient simulation in classical computers is therefore granted.

Defining the appropriate fitness function that will efficiently direct the search for the desired QECCs is crucial and not straightforward. The crux of the matter involves defining a phenotype set that captures the features expected of a good error correction code. For the toy problem, the definition of the problem itself contains the phenotype and leads to the form of the fitness function. Backed by the content developed in Chapter 3, we now present metrics capable of measuring the effectiveness of a circuit in correcting quantum errors and a method of evaluating them.

4.3.1 QECC fitness function

One can represent a given QECC by a set of two or more mutually orthogonal codewords $\{|c_i\rangle\}$. The codewords define a set of syndrome operators $\{S_i\}$ employed to detect and correct up to t errors following the scheme shown in Figure 3.4. With the syndrome operators and the subset of errors $\mathcal{E}$ we require our code to be able to detect and correct, one builds the syndrome table. One may then use it to decide if the set of codewords forms a functional QECC by checking if there are undetectable and uncorrectable errors. We reiterate how to classify and account for these errors:

- syndromes represented by bit strings of 0-s correspond to undetectable errors. Thus, given the syndrome table, the number of undetectable errors is obtained by counting how many errors return 0-stringed syndromes;
- if more than one error is associated with distinctive syndromes, all 2-on-2 combinations are cross-checked with the shared stabilizers of all codewords. If one combination fails, we classify the errors associated to the syndrome as uncorrectable.

Let e_{und} and e_{unc} be the number of undetectable and uncorrectable errors associated to a set of codewords, respectively. Then, the *corrigibility degree* c_d is given by

$$c_d \equiv (|\mathcal{E}| - e_{\text{und}} - e_{\text{unc}})/|\mathcal{E}| \tag{4-7}$$

We use the corrigibility degree as our main phenotype to evolve good QECCs, however its evaluation assumes the possession of a tentative set of codewords. How do we relate the Clifford circuits we employ in the GA with codewords? Taking Shor's code as an illustrative example, we may represent it by its codewords or by its EC as depicted in Figure 3.5. It then becomes natural to evolve ECs for our purposes, but there are some caveats. The EC in itself is insufficient to determine the codewords; one must know which initial states it acts to form them. For the particular circuit we portrayed for Shor's code, the $|\psi\rangle$ ket in the first register implicitly informs that the initial states are taken to be $\{|00\ldots0\rangle, |10\ldots0\rangle\}$. Furthermore, other circuits generate equivalent codewords for Shor's code. For example, consider the circuit shown in Figure 4.11. If it acts on the initial states $\{|0\ldots000\rangle, |0\ldots010\rangle\}$, it effectively generates equivalent codewords to (3-27) and (3-28)[3].

The main takeaway is that if we are going to evolve ECs, we need an algorithm capable of producing different sets of codewords from various

[3]The sets of codewords are in themselves different. The equivalence is in the sense that both sets work in a similar fashion forming a $[[9, 1, 3]]$ QECC.

combinations of initial states. For instance, if the GA evolved the circuit from Figure 4.11 and only tried to use $\{|00\ldots0\rangle, |10\ldots0\rangle\}$ as the initial states to form the codewords, it would wrongly conclude that the circuit does not constitute a good QECC.

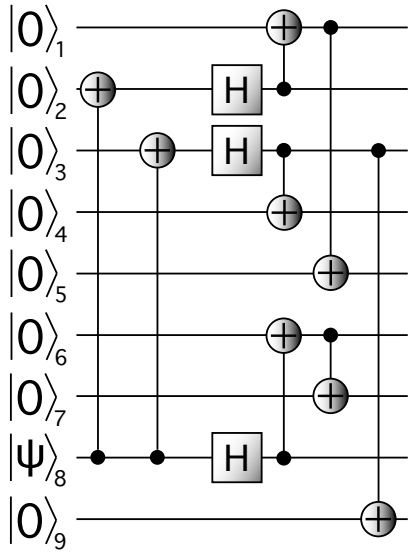

Figure 4.11: An example of an equivalent EC to Shor's error correction code.

We now describe the procedure we built to evaluate different sets of codewords given a Clifford circuit regarded as a tentative EC for a QECC. Also, the process captures the code *distance*, which we use as part of the QECC phenotype, as will be discussed.

Given an EC, applying it to the $|0\rangle^{\otimes n}$ state generates a first potential codeword $|c_0\rangle$. Recollecting that codewords form a set of mutually orthogonal states, we create a method to build $2^n - 1$ mutually orthogonal states to $|c_0\rangle$ which will give us a set of 2^n potential codewords[4] $\{|c_i\rangle\}$. In possession of $\{|c_i\rangle\}$, it remains to evaluate the c_d of subsets. To generate states orthogonal to $|c_0\rangle$, we find a set of logical $\mathcal{X} \equiv \{\bar{X}_i\}$ operators such that

$$\bar{X}_i |c_0\rangle = |c_i\rangle \tag{4-8}$$

$$\langle c_i|c_j\rangle = \delta_{ij} \tag{4-9}$$

$\forall i \in \{1, \ldots, 2^{n-1}\}$.

There is a systematic method to build a particular[5] set $\mathcal{X}$ that satisfies the above equations starting from the computational basis. Consider then the n-qubit computational basis. Starting from $|\psi_{0,I}\rangle = |0\rangle^{\otimes n}$ [6], it is straightfor-

[4]Since we work with qubits, a n-dimensional Hilbert space is spawned by 2^n states.

[5]$\mathcal{X}$ is not unique.

[6]The first subscript 0 refers to $|0\rangle^{\otimes n}$ and the I subscript stands for the identity in the sense that no computation was done.

ward to verify that the logical $\{\bar{X}_{i,I}\}$ operators that takes $|\psi_{0,I}\rangle$ to the other states of the basis $|\psi_{i,I}\rangle$ (which are mutually orthogonal by definition) are all the $2^n - 1$ tensor product combinations of Pauli letters X and I possessing at least one X. Define $[\bar{X}_I]$ as a $2^n - 1 \times n$ matrix whose rows are $\{\bar{X}_{i,I}\}$:

$$[\bar{X}_I] = \begin{bmatrix} X & I & I & \dots & I & I \\ I & X & I & \dots & I & I \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ I & I & I & \dots & I & X \\ X & X & I & \dots & I & I \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ X & X & X & \dots & X & X \end{bmatrix}. \tag{4-10}$$

Notice that $\{\bar{X}_{i,I}\}$ satisfies Equations (4-8) and (4-9). Let $|\psi_{0,U}\rangle = U|\psi_{0,I}\rangle$, where U is some unitary computation. U transforms each $\bar{X}_{i,I}$ into $\bar{X}_{i,U} \equiv U\bar{X}_{i,I}U^\dagger$ such that

$$\begin{aligned} \langle\psi_{0,U}|\,\bar{X}_{i,U}\,|\psi_{0,U}\rangle &= \langle\psi_{0,I}|\,U^\dagger U\bar{X}_{i,I}U^\dagger U\,|\psi_{0,I}\rangle \\ &= \langle\psi_{0,I}|\,\bar{X}_{i,I}\,|\psi_{0,I}\rangle \\ &= 0 \end{aligned} \tag{4-11}$$

$\forall i$. Each $\bar{X}_{i,U}$ is distinct since if $\bar{X}_{i,U} = \bar{X}_{j,U}$ for some i, j , then

$$U\left(\bar{X}_{i,I} - \bar{X}_{j,I}\right)U^\dagger = 0. \tag{4-12}$$

Since $U \neq 0$ and $\{\bar{X}_{i,I}\}$ are different by construction, then forcibly $i = j$. We conclude that $\{\bar{X}_{i,U}\}$ forms a set of logical operators whose elements take $|\psi_{0,U}\rangle$ into $2^n - 1$ unique mutually orthogonal states $|\psi_{i,U}\rangle$. Taking the particular case of U as an EC, $\mathcal{X} = \{\bar{X}_{i,\mathrm{EC}}\}$ is the set we seek.

Notice the similarity of the above arguments and the stabilizer formalism, where we also consider conjugations of the form $U \cdot U^\dagger$ for similar reasons. In addition, $[\bar{X}_I]$ is analogous to the tableau representation introduced in Section 2.2, which leads to the method we employ to evaluate $\{\bar{X}_i\}$: given an EC, we evaluate $U[\bar{X}_I]U^\dagger$ row by row using the conjugation rules on Table 2.2. One could skip the evaluation of $\mathcal{X}$ and directly assess $\{|c_i\rangle\}$ by applying the EC to each state of the computational basis. While this is true, having $\mathcal{X}$ provides the bit-flip distance between $|c_0\rangle \equiv |\psi_{0,\mathrm{EC}}\rangle$ and each $|c_i\rangle \equiv |\psi_{i,\mathrm{EC}}\rangle$, an important piece of information we use to speed up the evaluation of the EC as follows.

There is a significant computational cost issue with the intent of evaluating the c_d for each subset of $\{|c_i\rangle\}$ due to the sheer amount of subsets. Therefore, we are forced to simplify our approach by considering a limited

number of subsets. First, we limited ourselves to evolving QECCs with two-dimensional code spaces which leads to an $O\left(2^{2n-1}\right)$ of subsets to consider for each tentative EC (a significant but not sufficient reduction). Second, by appealing to symmetry, we make the heuristic argument that we can fix one of the codewords, as $|c_0\rangle$ without loss, and only consider subsets of the form $\{|c_0\rangle, |c_i\rangle\}$ for $i = 1, \ldots, 2^n - 1$. Finally, we further reduce the list of subsets by only evaluating the c_d for codewords with large and *balanced* code distances. For each pair $\{|c_0\rangle, |c_i\rangle\}$, the code distance is $\min(\mathrm{d_{b,i}}, \mathrm{d_{p,i}})$, where $d_{b,i}$ ($d_{p,i}$) is the bit-flip (phase-flip) distance. We only evaluate the c_d for pairs that maximize

$$\frac{\left(d_{b,i} d_{p,i}\right)^2}{\sqrt{|d_{b,i} - d_{p,i}|} + 1} \tag{4-13}$$

in relation to the set of $2^n - 1$ tentative pairs. This condition gives preference to pairs with larger code distance without introducing a bias towards bit or phase-flip errors. As noted, the calculation of $\mathcal{X}$ gives us d_b; so it remains to evaluate d_p. Logical phase-flip operators between a pair of orthogonal stabilizer states are Pauli words that stabilize both states but with different overall signs. Therefore, given $\{|c_0\rangle, |c_i\rangle\}$, d_p is given by the minimal-weight operator in the set $\mathcal{S}(|c_0\rangle) \cap \mathcal{S}(|c_i\rangle)$.

We are now ready to put together our fitness function. Consider a tentative EC with circuit depth D. Let $V_\perp$ be the set of all codewords $\{|c_0\rangle, |c_i\rangle\}$ (built as explained above) with associated $c_{d,i}$, $d_{b,i}$ and $d_{p,i}$. The fitness associated to the EC is then given by

$$\mathcal{F} = \max_{V_\perp} \; F \tag{4-14}$$

where

$$F = (1 + c_{d,i})^{10} + \frac{1}{D} + \min(\mathrm{d_{b,i}}, \mathrm{d_{p,i}}). \tag{4-15}$$

The form of Equation (4-15) is the result of attempts among other alternatives. For instance, we tested and excluded functional forms containing a term proportional to c_d/D, as we observed a tendency to evolve circuits with low depth to the detriment of c_d which we imperatively wish to be unity. A simple solution is to give greater weight to the phenotypes that we consider most important — in our case incorporating c_d as $(1 + c_d)^{10}$, where the power factor was chosen so that changes to c_d would be more meaningful than changes in D. The last term of (4-15) gives a bias towards QECCs with bigger distance.

4.3.2 Results

We have applied the GA to the problem of searching QECCs correcting up to single-qubit errors using the fitness function as defined in Equations

(4-14) and (4-15). We tested the GA performance against RS for a range of qubit overheads, from $n = 5$ to $n = 10$. Figure 4.12 displays the results for each case.

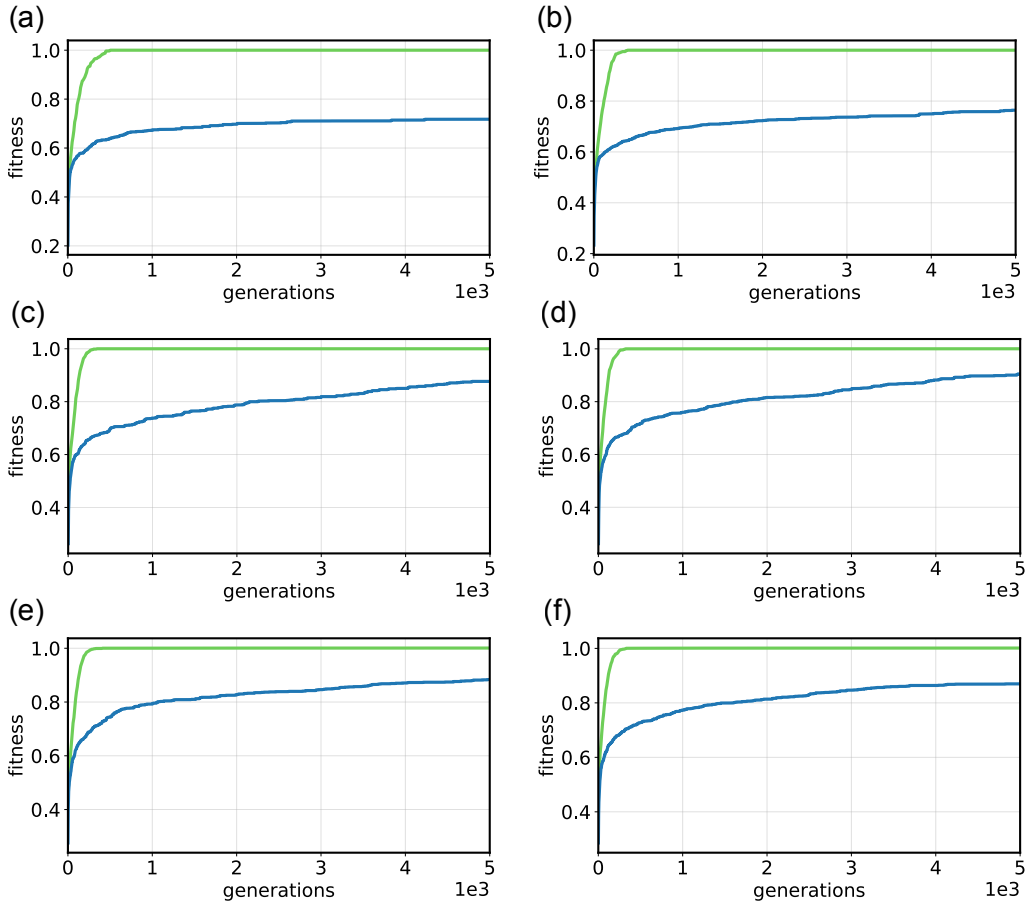

Figure 4.12: Evolutionary search through a genetic algorithm (green line) versus random search (blue line) in search of QECCs. Each curve is the average over 100 runs of the population's best fitness. The fitness values are normalized. (a) $n = 5$; (b) $n = 6$; (c) $n = 7$; (d) $n = 8$; (e) $n = 9$; (f) $n = 10$.

In all six cases, the GA showed a clear advantage over RS by being able to evolve functional, low-depth QECCs in just a few hundred generations. Indeed, the GA was capable of finding the perfect 5-qubit and Shor's 9-qubit code many times starting from random circuits and with no insight into these specific codes.

An unexpected result occurred: both the GA and the RS performed better for increasing n (it is harder to see for the GA, but there is a slight increase for each higher value of n). Recalling that the number of pure stabilizer states scales as $O(2^{n^2})$ [37], we anticipated it to be easier to find codes with a fewer number of qubits, e.g., $n = 5$. Observing the plots in Figure 4.12, we see the opposite behavior, which makes us hypothesize the reason.

Inequality (3-35), demonstrated in Section 3.2.3, shows that the number of single-qubit errors grows linearly while the number of available potential syndrome codes grows exponentially with n. Moreover, this inequality restricts itself to non-degenerate codes. Notice that the GA/RS also searches for degenerate codes which use less than $s(n,t)+1$ syndrome codes. Finally, recollecting the topic of equivalent ECs, one of its causes is the freedom to exchange registers. Figure 4.13 illustrates this point where any register permutation creates a new equivalent circuit. Since the number of permutations grows as $n!$, the number of available code syndromes as 2^{n-1}, and we consider degenerate codes as well, we conjecture that the number of possible QECCs rapidly grows as n gets larger making it easier to find them.

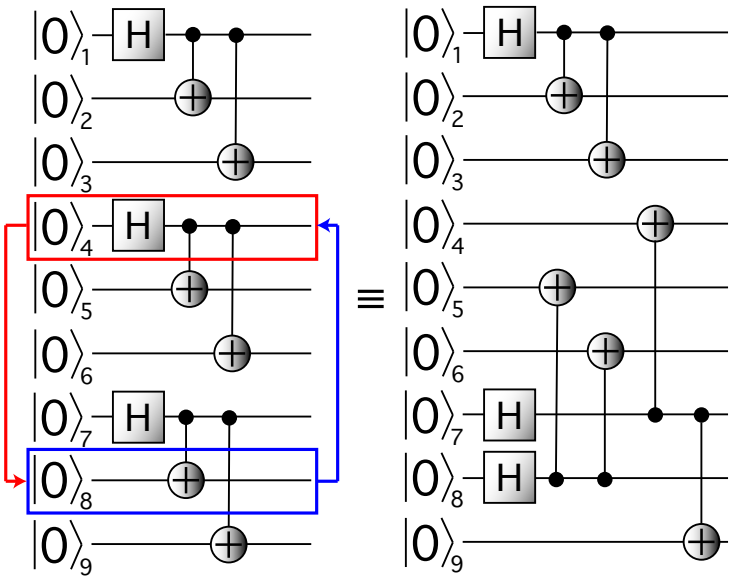

Figure 4.13: Creation of an equivalent EC for Shor's code by permutation of two qubit registers. Any permutation of the register produces an equivalent EC.

Nevertheless, RS seldom found a solution within the limit of $5,000$ generations, while the GA consistently evolved good QECC[7]. Table 4.1 shows how many times each method successfully found a solution within 100 trials for each value of n tested. While the GA had a success rate of 100% in all cases (ignoring, for the moment, the color case), RS varied between 4% and 40% showing an abrupt downward trend from $n=8$. Therefore, even though it appears from the graphs in Figure 4.12 that the RS is getting more efficient as n gets larger, the data in Table 4.1 suggests otherwise. Perhaps, for even larger n, RS becomes increasingly more inefficient. Further, the difference in the speed at which the methods converge to the solution is undeniable. The

[7]We consider that a QECC was found if an EC with $c_d = 1$ was generated.

Table 4.1: Number of times each method found a solution in 100 trials with a maximum limit of $5,000$ generations (except for the color case, in which the limit was $10,000$ generations). We divide the results into the cases where the solution found has a code distance greater or equal to 3 and in which it is greater than 3.

n	GA	RS	GA	RS
	dist. ≥ 3		dist. > 3	
5	100	10	0	0
6	100	4	0	0
7	100	40	0	0
8	100	32	31	0
9	100	13	47	0
10	100	10	73	1
7 (color)	54	0	NA	NA

speed of convergence becomes increasingly important for larger n since the computation costs scale.

In addition to striving to verify the practicality of GAs by showing their advantage over RS, we uttermost want to show (1) its flexibility to find surprising results and (2) its applicability in specific problems. For (1), the GA evolved unexpected QECCs in many instances. For example, for $n = 9$ where we expect to generate Shor's code as a benchmark, the GA many times produced $[[9, 1, 5]]$ codes with a striking code distance of 5. Figure 4.14 shows an example of a $[[9, 1, 5]]$ evolved by our GA. These circuits have a larger distance than Shor's code, guaranteeing a higher fitness at the expense of a higher circuit depth. Additionally, for $n = 10$, many $[[10, 1, 5]]$ codes were also evolved. In contrast, the RS found a code with a distance greater than 3 only once. Figure 4.15 show an example of a $[[10, 1, 5]]$ code.

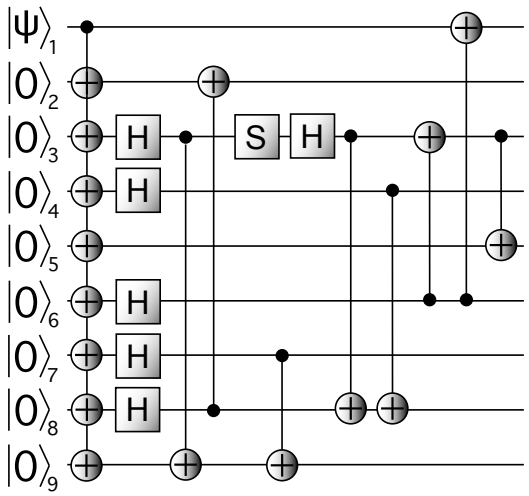

Figure 4.14: Example of a $[[9, 1, 5]]$ EC found by GA with depth $D = 8$.

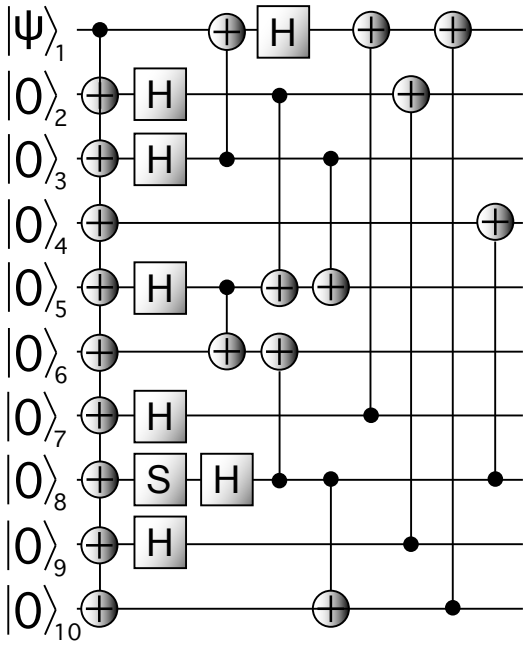

Figure 4.15: Example of a $[[10, 1, 5]]$ EC found by GA with depth $D = 6$.

For (2), we decided to tweak the fitness function to specialize the GA in searching color codes [43]. Color codes are a particular class of error-correcting codes belonging to the broader family of *topological* QECCs [43, 88]. The main feature of topological codes is their modularity, i.e., the main code is assembled by patching elementary repeated pieces. Modularity makes scaling up circuits straightforward, and each module requires only nearest-neighbor interactions which relieves a major hardware constraint. These reasons place topological codes among the leading prospects for actual hardware implementation [89–91].

In particular, we set out to evolve two-dimensional[8] color codes since a simple modification in the fitness function is required to guide evolution towards them. Two-dimensional color codes are also members of the CSS class of QECCs [42, 61]. Given a set of codewords forming a QECC, if their common stabilizer set can be generated by stabilizers made only by Xs or Zs independently (each generator is made only by Xs or Zs), then it is a CSS code [43, 62, 63]. We then build a method to quantify how well a code meets this requirement in a similar fashion to the corrigibility degree. Given a set of codewords, we measure its CSS degree $\mathrm{CSS_d}$ by constructing a generator set, for their joint stabilizers, with the maximum number of operators made only by Xs or Zs possible. We define $\mathrm{CSS_d}$ as the ratio of operators that satisfies the CSS criterion by the total number of elements in the generator set. Hence, to drive the GA towards two-dimensional color codes, we modify Eq. (4-15) to

[8]Two-dimensional in the sense of the dimensionality of the lattice formed by the qubits' interactions, not the Hilbert space code dimension.

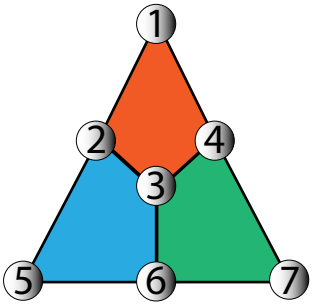

Figure 4.16: A lattice arrangement of the 7-qubit color code.

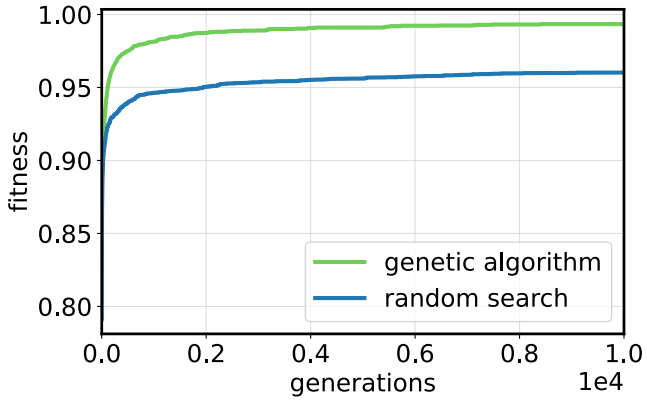

Figure 4.17: Evolutionary search through a genetic algorithm versus random search in search for the 7-qubit color code. Each curve is the average over 100 runs of the population's best fitness. The fitness values are normalized.

$$F = (1 + c_{d,i})^{10} + (1 + \mathrm{CSS}_{\mathrm{d,i}})^{10} + \frac{1}{D} + \min(\mathrm{d}_{\mathrm{b,i}}, \mathrm{d}_{\mathrm{p,i}}). \qquad (4\text{-}16)$$

Again, we use form $(1 + \mathrm{CSS}_{\mathrm{d,i}})^{10}$ so that the GA focuses on evolving codes that satisfies the CSS criterion.

The simplest two-dimensional color code is the triangular 7-qubit code [43]. Its name derives from the fact that one can represent its topology with a triangular lattice [92] made by three plaquettes as illustrated in Figure 4.16. Hence, we applied the GA in search for the 7-qubit code using the updated fitness function (4-16). The plot of figure 4.17 shows the performance of the GA against RS. The fitness values are normalized by a fitness-target value taken from an EC found in [35]. Noting the greater difficulty in evolving the color code, we increased the generation limit to $10,000$. It appears again that RS performed reasonably well compared to the GA, but again the performance plot can be deceiving — as shown in Table 4.1, while the GA had a success rate of 54%, RS never found the color code.

5
Outlook

In this work, we set out to reveal the potential of evolutionary algorithms in developing quantum algorithms through proof-of-concept examples, notably the evolution of stabilizer quantum error correction codes. In our evolution simulations, stabilizer states became the individuals, their underlying circuits stood for digital genotypes. We designed genetic operators inspired by the circuit's structure and set up mating selection and reproduction rules. Thus, covering the main aspects that drive biological evolution and particularities brought by quantum circuits, we conceived Algorithm 5 — the backbone of our quantum GA.

By devising a suitable fitness function, the GA proved capable of evolving Shor's 9-qubit QECC, and the 5-qubit perfect code repeatedly from random circuits. Furthermore, recollecting the results shown in the plots of Figure 4.12, the GA significantly outperformed RS in all instances by converging to a solution on a much shorter time scale. While RS aimlessly samples through the whole space of possible genotypes, the GA is bound to search over a subspace that lies within the neighborhood of a given genotype. The subspace sampled by the GA is then significantly smaller than that of RS, while the fitness function introduces a directed search that clearly offers an edge over blind sampling.

In addition to evolving already known codes, other competing solutions have emerged as, e.g., the circuits depicted in Figures 4.14 and 4.15. This indicates that the sample space is likely rich with contending solutions, each with distinct features that we may further sort out by altering the fitness function according to our needs. Indeed, we showcased an example of this approach by specializing the QECC fitness function in evolving CSS codes. With the tweaked fitness and searching for 7-qubits codes, we evolved several times the triangular 7-qubit color code, a topological QECC.

Future developments

We anticipate at least three directions of future research in which ideas closely related to the ones introduced here might lead to interesting

developments within the broader field of quantum information and computing.

First, evolution might offer valuable means to search for other topological codes. It might be possible to encode topological features of quantum states into an appropriate fitness function, thus enabling the automated search for new topologically ordered states characterized by distinct values of the topological entanglement entropy [93, 94].

Second, quantum algorithm compilation [95] is a challenging problem that will become increasingly important as noisy intermediate-scale quantum (NISQ) computers start to emerge [44,96]. Hence, the development of quantum compiling methods is a very active, timely research field [96–104]. In principle, the GA seems a promising platform to automate quantum compiling and depth reduction. For example, given a target algorithm we wish to compile, we could use a fidelity measurement between the algorithm and a tentative circuit times the inverse of its depth as a fitness function.

Third, one may translate quantum hardware specifications into metrics incorporated in the fitness function to produce tailor-made algorithms. As an illustration, Figure 5.1 shows the available lattices of the quantum system devices IBM offers on their IBM Quantum service [105]. Note that each device geometry imposes interaction restrictions on two-qubit gates. Given a device with a particular lattice, we could incorporate a penalty factor for circuits that disobey the geometry restrictions. Another possible approach could be to prohibit two-qubit gates between non-neighbor qubits[1]. Additionally, some gates may be harder to implement due to the experimental setup. Brown and Susskind in [106] introduce the concept of *unitary complexity* quantifying how hard it is to implement a given unitary transformation given the properties of the setup. The unitary complexity could also be used as a penalty factor in the fitness function to penalize circuits possessing complex gates for a given hardware.

Final remarks

When we started this project, we had the faintest idea if an evolutionary algorithm would succeed in evolving quantum circuits. We find it remarkable how the GA we built, with a few hundred lines of code, can evolve in minutes celebrated results that needed the minds of Raymond Laflamme, Peter Shor, and Robert Calderbank to come about. The success of the GA makes us wonder about all the possibilities it brings. As quantum computers improve

[1]Maybe this imposes too big a constraint leading the GA to have difficulties finding a solution.

with countless qubits in ever-larger and more complex topologies, genetic algorithms may yet prove to be an invaluable optimization method in the quantum computing engineer's toolbox.

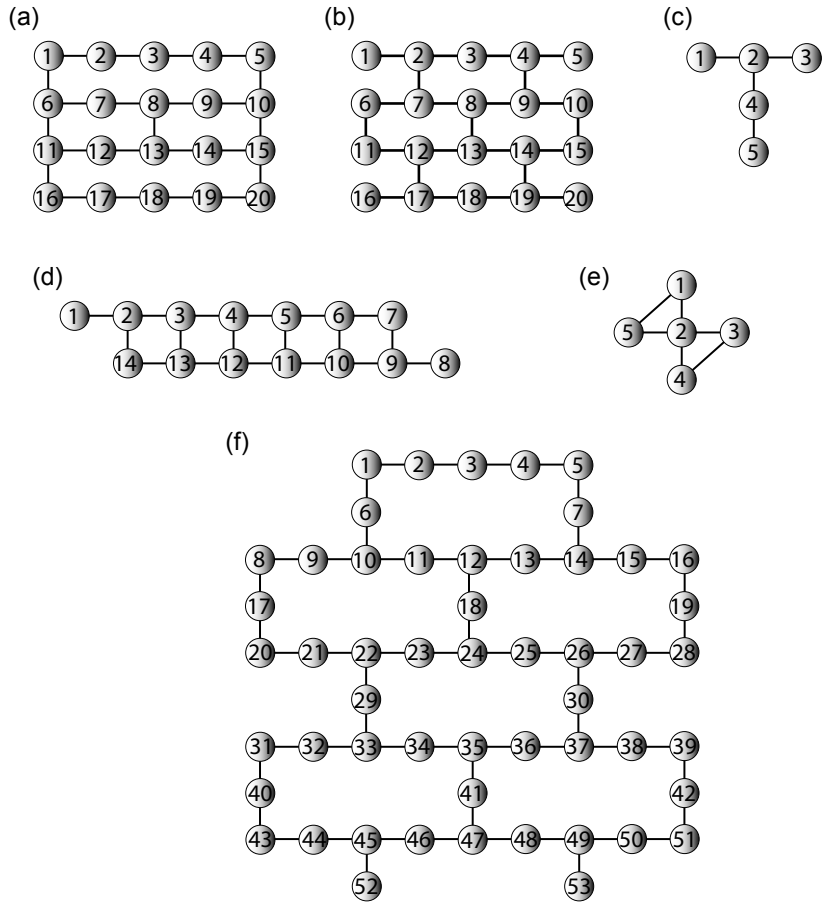

Figure 5.1: IBM quantum systems lattices taken from [105]. The systems are named: (a) 20-qubits systems Johannesburg and Poughkeepsie; (b) 20-qubits systems Almaden, Boeblingen, and Singapor; (c) 5-qubits systems Ourense, Valencia, and Vigo; (d) 14-qubits system Melbourne; (e) 5-qubits system Yorktown; (f) 53-qubits system Rochester.

Bibliography

[1] BIALEK, W.. **Biophysics: Searching for Principles**. Princeton University Press, 2012.

[2] KAUFFMAN, S. A.. **World Beyond Physics: The Emergence and Evolution of Life**. Oxford University Press, USA, 2019.

[3] JOANA C. XAVIER, E. A.. **Autocatalytic chemical networks at theorigin of metabolism**. Proc. R. Soc. B, 287:20192377., 2020.

[4] BRYAN LAU, E. A.. **An introduction to ratchets in chemistry and biology**. Materials Horizons, 2017.

[5] ALBERTS, B.; BRAY, D.; HOPKIN, K.; JOHNSON, A. D.; LEWIS, J.; RAFF, M.; ROBERTS, K. ; WALTER, P.. **Essential cell biology**. Garland Science, 2015.

[6] KAUFFMAN, S.; ROLI, A.. **The world is not a theorem**. Entropy, 23(11), 2021.

[7] KIRYL D. PIATKEVICH, E. A.. **A robotic multidimensional directed evolution approach applied to fluorescent voltage reporters**. Nat Chem Biol., 14(4):352, 2018.

[8] SHOR, P. W.. **Why haven't more quantum algorithms been found?** Journal of the ACM, 50:87, 2003.

[9] MARTYN, J. M.; ROSSI, Z. M.; TAN, A. K. ; CHUANG, I. L.. **Grand unification of quantum algorithms**. PRX Quantum, 2:040203, Dec 2021.

[10] NIELSEN, M. A.; CHUANG, I.. **Quantum computation and quantum information**. American Association of Physics Teachers, 2002.

[11] BRAVYI, S.; GOSSET, D. ; KÖNIG, R.. **Quantum advantage with shallow circuits**. Science, 362(6412):308–311, 2018.

[12] ITEN, R.; METGER, T.; WILMING, H.; DEL RIO, L. ; RENNER, R.. **Discovering physical concepts with neural networks**. Phys. Rev. Lett., 124:010508, Jan 2020.

[13] MILES CRANMER, E. A.. **Discovering symbolic models from deep learning with inductive biases**. arXiv:2006.11287, 2020.

[14] MILES CRANMER, E. A.. **Lagrangian neural networks**. arXiv:2003.04630, 2020.

[15] JARED O'LEARY, JOEL A. PAULSON, A. M.. **Stochastic physics-informed neural networks (spinn): A moment-matching framework for learning hidden physics within stochastic differential equations**. arXiv:2109.01621, 2021.

[16] KRENN, M.; MALIK, M.; FICKLER, R.; LAPKIEWICZ, R. ; ZEILINGER, A.. **Automated search for new quantum experiments**. Phys. Rev. Lett., 116:090405, Mar 2016.

[17] JUAN MIGUEL ARRAZOLA, E. A.. **Machine learning method for state preparation and gate synthesis on photonic quantum computers**. Quantum Science and Technology, 4:024004, 2019.

[18] MARIO KRENN, MANUEL ERHARD, A. Z.. **Computer-inspired quantum experiments**. Nature Reviews Physics, 2:649, 2020.

[19] KNOTT, P. A.. **A search algorithm for quantum state engineering and metrology**. New Journal of Physics, 18(7):073033, jul 2016.

[20] ROSANNA NICHOLS, E. A.. **Designing quantum experiments with a genetic algorithm**. Quantum Sci. Technol., 4:045012, 2019.

[21] RAMBHATLA, K.; D'AURELIO, S. E.; VALERI, M.; POLINO, E.; SPAGNOLO, N. ; SCIARRINO, F.. **Adaptive phase estimation through a genetic algorithm**. Phys. Rev. Research, 2:033078, Jul 2020.

[22] CARLEO, G.; TROYER, M.. **Solving the quantum many-body problem with artificial neural networks**. Science, 355(6325), 2017.

[23] GAO, J.; QIAO, L.-F.; JIAO, Z.-Q.; MA, Y.-C.; HU, C.-Q.; REN, R.-J.; YANG, A.-L.; TANG, H.; YUNG, M.-H. ; JIN, X.-M.. **Experimental machine learning of quantum states**. Phys. Rev. Lett., 120:240501, Jun 2018.

[24] CANABARRO, A.; BRITO, S. ; CHAVES, R.. **Machine learning nonlocal correlations**. Phys. Rev. Lett., 122:200401, May 2019.

[25] KRIVÁCHY, T.; CAI, Y.; CAVALCANTI, D.; TAVAKOLI, A.; GISIN, N. ; BRUNNER, N.. **A neural network oracle for quantum nonlocality problems in networks**. npj Quantum Information, 6(1):1–7, 2020.

[26] AGNIESZKA WOŁOS, E. A.. **Synthetic connectivity, emergence, and self-regeneration in the network of prebiotic chemistry**. Science, 369:1584, 2020.

[27] REAL, E.; AGGARWAL, A.; HUANG, Y. ; LE, Q. V.. **Regularized evolution for image classifier architecture search**. In: PROCEEDINGS OF THE AAAI CONFERENCE ON ARTIFICIAL INTELLIGENCE, volumen 33, p. 4780–4789, 2019.

[28] OLSSON, C.; BHUPATIRAJU, S.; BROWN, T.; ODENA, A. ; GOODFELLOW, I.. **Skill rating for generative models**. arXiv preprint arXiv:1808.04888, 2018.

[29] POULIN, D.; QARRY, A.; SOMMA, R. ; VERSTRAETE, F.. **Quantum simulation of time-dependent hamiltonians and the convenient illusion of hilbert space**. Phys. Rev. Lett., 106:170501, Apr 2011.

[30] SUSSKIND, L.. **Three lectures on complexity and black holes**. arXiv:1810.11563, 2018.

[31] BRANDÃO, F. G.; CHEMISSANY, W.; HUNTER-JONES, N.; KUENG, R. ; PRESKILL, J.. **Models of quantum complexity growth**. PRX Quantum, 2:030316, Jul 2021.

[32] FRANK ARUTE, E. A.. **Quantum supremacy using a programmable superconducting processor**. Nature, 574:pages505, 2019.

[33] WU, Y.; BAO, W.-S.; CAO, S.; CHEN, F.; CHEN, M.-C.; CHEN, X.; CHUNG, T.-H.; DENG, H.; DU, Y.; FAN, D.; GONG, M.; GUO, C.; GUO, C.; GUO, S.; HAN, L.; HONG, L.; HUANG, H.-L.; HUO, Y.-H.; LI, L.; LI, N.; LI, S.; LI, Y.; LIANG, F.; LIN, C.; LIN, J.; QIAN, H.; QIAO, D.; RONG, H.; SU, H.; SUN, L.; WANG, L.; WANG, S.; WU, D.; XU, Y.; YAN, K.; YANG, W.; YANG, Y.; YE, Y.; YIN, J.; YING, C.; YU, J.; ZHA, C.; ZHANG, C.; ZHANG, H.; ZHANG, K.; ZHANG, Y.; ZHAO, H.; ZHAO, Y.; ZHOU, L.; ZHU, Q.; LU, C.-Y.; PENG, C.-Z.; ZHU, X. ; PAN, J.-W.. **Strong quantum computational advantage using a superconducting quantum processor**. Phys. Rev. Lett., 127:180501, Oct 2021.

[34] JERRY CHOW, OLIVER DIAL, J. G.. **Ibm quantum breaks the 100 qubit barrier**. Technical report, IBM, https://research.ibm.com/blog/127-qubit-quantum-processor-eagle, 2021.

[35] LUKAS POSTLER, E. A.. **Demonstration of fault-tolerant universal quantum gate operations**. arXiv:2111.12654, 2021.

[36] GOTTESMAN, D.. **Stabilizer codes and quantum error correction**. PhD thesis, California Institute of Technology, 1997.

[37] AARONSON, S.; GOTTESMAN, D.. **Improved simulation of stabilizer circuits**. Phys. Rev. A, 70:052328, Nov 2004.

[38] GIDNEY, C.. **Stim: a fast stabilizer circuit simulator**. Quantum, 5:497, 2021.

[39] ROFFE, J.. **Quantum error correction: an introductory guide**. Contemporary Physics, 60(3):226–245, 2019.

[40] LAFLAMME, R.; MIQUEL, C.; PAZ, J. P. ; ZUREK, W. H.. **Perfect quantum error correcting code**. Physical Review Letters, 77(1):198, 1996.

[41] SHOR, P. W.. **Scheme for reducing decoherence in quantum computer memory**. Physical review A, 52, Oct 1995.

[42] CALDERBANK, A. R.; RAINS, E. M.; SHOR, P. W. ; SLOANE, N. J.. **Quantum error correction and orthogonal geometry**. Physical Review Letters, 78(3):405, 1997.

[43] KUBICA, A. M.. **The ABCs of the color code: A study of topological quantum codes as toy models for fault-tolerant quantum computation and quantum phases of matter**. PhD thesis, California Institute of Technology, 2018.

[44] PRESKILL, J.. **Quantum computing in the nisq era and beyond**. Quantum, 2:79, 2018.

[45] LI, Y.; FISHER, M. P.. **Statistical mechanics of quantum error correcting codes**. Physical Review B, 103(10):104306, 2021.

[46] NIELSEN, M. A.. **A geometric approach to quantum circuit lower bounds**. arXiv preprint quant-ph/0502070, 2005.

[47] BROWN, A. R.; SUSSKIND, L.. **Complexity geometry of a single qubit**. Physical Review D, 100(4):046020, 2019.

[48] DEKEL, E.; ALON, U.. **Optimality and evolutionary tuning of the expression level of a protein**. Nature, 436(7050):588–592, 2005.

[49] KIANI, B. T.; LLOYD, S. ; MAITY, R.. **Learning unitaries by gradient descent**. arXiv preprint arXiv:2001.11897, 2020.

[50] HEYFRON, L. E.; CAMPBELL, E. T.. **An efficient quantum compiler that reduces t count**. Quantum Science and Technology, 4(1):015004, 2018.

[51] NAM, Y.; ROSS, N. J.; SU, Y.; CHILDS, A. M. ; MASLOV, D.. **Automated optimization of large quantum circuits with continuous parameters**. npj Quantum Information, 4(1):1–12, 2018.

[52] JONES, T.; BENJAMIN, S. C.. **Robust quantum compilation and circuit optimisation via energy minimisation**. Quantum, 6:628, 2022.

[53] TANDEITNIK, D.. **Evolving quantum circuits**. `https://github.com/tandeitnik/Evolving_Quantum_Circuits`, 2022.

[54] GOTTESMAN, D.. **The heisenberg representation of quantum computers**. arXiv preprint quant-ph/9807006, 1998.

[55] GARCIA, H. J.; MARKOV, I. L. ; CROSS, A. W.. **Efficient inner-product algorithm for stabilizer states**. arXiv preprint arXiv:1210.6646, 2012.

[56] NAHUM, A.; RUHMAN, J.; VIJAY, S. ; HAAH, J.. **Quantum entanglement growth under random unitary dynamics**. Phys. Rev. X, 7:031016, Jul 2017.

[57] RALPH, T.. **Quantum error correction of continuous-variable states against gaussian noise**. Physical Review A, 84(2):022339, 2011.

[58] DIAS, J.; RALPH, T. C.. **Quantum error correction of continuous-variable states with realistic resources**. Physical Review A, 97(3):032335, 2018.

[59] WOOTTERS, W. K.; ZUREK, W. H.. **A single quantum cannot be cloned**. Nature, 299(5886):802–803, 1982.

[60] GREENBERGER D.M., HORNE M.A., Z. A.. **Going beyond bell's theorem**. In: M., K., editor, BELL'S THEOREM, QUANTUM THEORY

AND CONCEPTIONS OF THE UNIVERSE, chapter 10, p. 69–72. Springer, Dordrecht, 1989.

[61] GOTTESMAN, D.. **Class of quantum error-correcting codes saturating the quantum hamming bound**. Physical Review A, 54(3):1862, 1996.

[62] CALDERBANK, A. R.; SHOR, P. W.. **Good quantum error-correcting codes exist**. Physical Review A, 54(2):1098, 1996.

[63] STEANE, A.. **Multiple-particle interference and quantum error correction**. Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences, 452(1954):2551–2577, 1996.

[64] DENNIS, E.; KITAEV, A.; LANDAHL, A. ; PRESKILL, J.. **Topological quantum memory**. Journal of Mathematical Physics, 43(9):4452–4505, 2002.

[65] DODD, M. S.; PAPINEAU, D.; GRENNE, T.; SLACK, J. F.; RITTNER, M.; PIRAJNO, F.; O'NEIL, J. ; LITTLE, C. T.. **Evidence for early life in earth's oldest hydrothermal vent precipitates**. Nature, 543(7643):60–64, 2017.

[66] DARWIN, C.. **On the origin of species, 1859**. Routledge, 2004.

[67] FOGEL, D. B.. **An introduction to simulated evolutionary optimization**. IEEE transactions on neural networks, 5(1):3–14, 1994.

[68] FRASER, A. S.. **Simulation of genetic systems by automatic digital computers i. introduction**. Australian journal of biological sciences, 10(4):484–491, 1957.

[69] BARKER, J.. **Simulation of genetic systems by automatic digital computers**. Australian Journal of Biological Sciences, 11(4):603–612, 1958.

[70] BREMERMANN, H. J.; OTHERS. **Optimization through evolution and recombination**. Self-organizing systems, 93:106, 1962.

[71] BREMERMANN, H. J.. **Numerical optimization procedures derived from biological evolution processes**. Cybernetic problems in bionics, p. 597–616, 1968.

[72] BREMERMANN, H. J.; ROGSON, M.. **An evolution-type search method for convex sets.** Technical report, CALIFORNIA UNIV BERKELEY, 1964.

[73] REED, J.; TOOMBS, R. ; BARRICELLI, N. A.. **Simulation of biological evolution and machine learning: I. selection of self-reproducing numeric patterns by data processing machines, effects of hereditary control, mutation type and crossing**. Journal of theoretical biology, 17(3):319–342, 1967.

[74] SAMPSON, J. R.. **Adaptation in natural and artificial systems (john h. holland)**, 1976.

[75] BARTZ-BEIELSTEIN, T.; BRANKE, J.; MEHNEN, J. ; MERSMANN, O.. **Evolutionary algorithms**. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 4(3):178–195, 2014.

[76] LUSH, J. L.. **Animal breeding plans**. Read Books Ltd, 2013.

[77] WILCZYNSKI, J. Z.. **On the presumed darwinism of alberuni eight hundred years before darwin**. Isis, 50(4):459–466, 1959.

[78] PURUGGANAN, M. D.; FULLER, D. Q.. **The nature of selection during plant domestication**. Nature, 457(7231):843–848, 2009.

[79] ROTH, S. C.. **What is genomic medicine?** Journal of the Medical Library Association: JMLA, 107(3):442, 2019.

[80] SMITH, J. M.; SZATHMARY, E.. **The major transitions in evolution**. OUP Oxford, 1997.

[81] PIERCE, B. A.. **Genetics: a conceptual approach**. Macmillan, 2012.

[82] GAVRILETS, S.. **Fitness landscapes and the origin of species (MPB-41)**. Princeton University Press, 2004.

[83] KAUFFMAN, S.; LEVIN, S.. **Towards a general theory of adaptive walks on rugged landscapes**. Journal of theoretical Biology, 128(1):11–45, 1987.

[84] HOLLAND, J. H.. **Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence**. MIT press, 1992.

[85] EIBEN, A. E.; RAUE, P.-E. ; RUTTKAY, Z.. **Genetic algorithms with multi-parent recombination**. In: INTERNATIONAL CONFERENCE ON PARALLEL PROBLEM SOLVING FROM NATURE, p. 78–87. Springer, 1994.

[86] TING, C.-K.. **On the mean convergence time of multi-parent genetic algorithms without selection**. In: EUROPEAN CONFERENCE ON ARTIFICIAL LIFE, p. 403–412. Springer, 2005.

[87] GENERSCHWEFEL, H.. **Evolution and optimum seeking**. Sixth Generation Computer Technology Series, John Wiley and Sons, 1995.

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

[89] NICKERSON, N. H.; FITZSIMONS, J. F. ; BENJAMIN, S. C.. **Freely scalable quantum technologies using cells of 5-to-50 qubits with very lossy and noisy photonic links**. Physical Review X, 4(4):041041, 2014.

[90] SETE, E. A.; ZENG, W. J. ; RIGETTI, C. T.. **A functional architecture for scalable quantum computing**. In: 2016 IEEE INTERNATIONAL CONFERENCE ON REBOOTING COMPUTING (ICRC), p. 1–6. IEEE, 2016.

[91] O'GORMAN, J.; NICKERSON, N. H.; ROSS, P.; MORTON, J. J. ; BENJAMIN, S. C.. **A silicon-based surface code quantum computer**. npj Quantum Information, 2(1):1–14, 2016.

[92] BERMUDEZ, A.; XU, X.; NIGMATULLIN, R.; O'GORMAN, J.; NEGNEVITSKY, V.; SCHINDLER, P.; MONZ, T.; POSCHINGER, U.; HEMPEL, C.; HOME, J. ; OTHERS. **Assessing the progress of trapped-ion processors towards fault-tolerant quantum computation**. Physical Review X, 7(4):041061, 2017.

[93] KITAEV, A.; PRESKILL, J.. **Topological entanglement entropy**. Physical review letters, 96(11):110404, 2006.

[94] LEVIN, M.; WEN, X.-G.. **Detecting topological order in a ground state wave function**. Physical review letters, 96(11):110405, 2006.

[95] HARROW, A.. **Quantum compiling**. PhD thesis, Citeseer, 2001.

[96] KHATRI, S.; LAROSE, R.; POREMBA, A.; CINCIO, L.; SORNBORGER, A. T. ; COLES, P. J.. **Quantum-assisted quantum compiling**. Quantum, 3:140, 2019.

[97] VENTURELLI, D.; DO, M.; RIEFFEL, E. ; FRANK, J.. **Compiling quantum circuits to realistic hardware architectures using temporal planners**. Quantum Science and Technology, 3(2):025004, 2018.

[98] BOOTH, K. E.; DO, M.; BECK, J. C.; RIEFFEL, E.; VENTURELLI, D. ; FRANK, J.. **Comparing and integrating constraint programming and temporal planning for quantum circuit compilation**. In: TWENTY-EIGHTH INTERNATIONAL CONFERENCE ON AUTOMATED PLANNING AND SCHEDULING, 2018.

[99] CINCIO, L.; SUBAŞI, Y.; SORNBORGER, A. T. ; COLES, P. J.. **Learning the quantum algorithm for state overlap**. New Journal of Physics, 20(11):113022, 2018.

[100] MASLOV, D.; DUECK, G. W.; MILLER, D. M. ; NEGREVERGNE, C.. **Quantum circuit simplification and level compaction**. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 27(3):436–444, 2008.

[101] BOOTH JR, J.. **Quantum compiler optimizations**. arXiv preprint arXiv:1206.3348, 2012.

[102] CHONG, F. T.; FRANKLIN, D. ; MARTONOSI, M.. **Programming languages and compiler design for realistic quantum hardware**. Nature, 549(7671):180–187, 2017.

[103] HEYFRON, L. E.; CAMPBELL, E. T.. **An efficient quantum compiler that reduces t count**. Quantum Science and Technology, 4(1):015004, 2018.

[104] HÄNER, T.; STEIGER, D. S.; SVORE, K. ; TROYER, M.. **A software methodology for compiling quantum programs**. Quantum Science and Technology, 3(2):020501, 2018.

[105] MCCLURE, D.; GAMBETTA, J.. **Quantum computation center opens**. `https://www.ibm.com/blogs/research/2019/09/quantum-computation-center/`, 2019. [Online; accessed 9-April-2022].

[106] BROWN, A. R.; SUSSKIND, L.. **Complexity geometry of a single qubit**. Physical Review D, 100(4):046020, 2019.

[107] PINTER, C. C.. **A book of abstract algebra**. Courier Corporation, 2010.

A
List of publications

1. BRANDÃO, Igor; TANDEITNIK, Daniel; GUERREIRO, Thiago. **Coherent scattering-mediated correlations between levitated nanospheres**. Quantum Science and Technology, v. 6, n. 4, p. 045013, 2021.

2. BRANDÃO, Igor; TANDEITNIK, Daniel; GUERREIRO, Thiago. **QuGIT: a numerical toolbox for Gaussian quantum states**. arXiv preprint arXiv:2201.06368, 2022.

3. TANDEITNIK, Daniel; GUERREIRO, Thiago. **Evolving Quantum Circuits**. *In preparation.*

B
Notation convection

B.1
Tensor products and indexing

Throughout this work quantum states are represented using Dirac's bra-ket notation. For single-qubits, a two-state quantum system, we primarily use the computational basis $\{|0\rangle, |1\rangle\}$. Hence, a general qubit states is written as

$$\alpha |0\rangle + \beta |1\rangle , \tag{B-1}$$

where $|\alpha|^2 + |\beta|^2 = 1$ to guarantee normalization of the state.

For multi-qubit states, the tensor product symbol and the indexing of states will be often omitted for reading clarity. As an example, consider the tensor product state of two qubits $|0\rangle_1 \otimes |0\rangle_2$. Such a state may be written as

$$|0\rangle_1 \otimes |0\rangle_2 = |0\rangle |0\rangle = |00\rangle . \tag{B-2}$$

The order of the indices are always crescent from left to right, even for bras:

$$\langle 0|_1 \langle 0|_2 = \langle 00| . \tag{B-3}$$

The tensor product symbol will also be omitted for operators, but indices will always be shown when working with more than one qubit. Hence, an operator as $X_1 \otimes Z_2 \otimes X_3$ will be written as $X_1 Z_2 X_3$. The identity operator I will be omitted whenever possible. For instance, we express the two qubit operator $I_1 X_2$ as X_2 with the presence of I_1 implied. For n-qubits, we express the identity $I_1 I_2 \dots I_n$ as just I.

B.2
Pauli operators

We use two representations for Pauli operators in this work. The Pauli operators are symbolized by

$$\sigma_{x,i} = X_i,\ \sigma_{y,i} = Y_i,\ \sigma_{z,i} = Z_i, \tag{B-4}$$

where the Latin dummy index i represents the index of the affected qubit. When we use the notation $\sigma_{j,i}$, it is implied that $j \in \{x, y, z\}$. We interchange-

ably refer to single Pauli operators as *Pauli letters* and the tensor product of two or more Pauli letters as *Pauli words*. We may also use digits instead of letters for the σ notation:

$$\sigma_{x,i} = \sigma_{1,i},\ \sigma_{y,i} = \sigma_{2,i},\ \sigma_{z,i} = \sigma_{3,i}. \tag{B-5}$$

For example, using digits is more suitable when working with the Levi-Civita symbol.

C
Pauli operators, Pauli general group and some properties

Define $\{I, X, Y, Z\}$ as the two-dimensional Pauli matrices represented in the computational base $\{|0\rangle, |1\rangle\}$ as

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}. \tag{C-1}$$

We define the Pauli group for a single qubit $\mathcal{G}_1$ as the set

$$\mathcal{G}_1 = \{\pm I, \pm iI, \pm X, \pm iX, \pm Y, \pm iY, \pm Z, \pm iZ\}. \tag{C-2}$$

The general Pauli group $\mathcal{G}_n$ over n-qubits is defined as the set composed of all tensor product combinations of the elements of $\mathcal{G}_1$. Given an element $g \in \mathcal{G}_n$, we define its *weight* $|g|$ as the number of non-identity operators composing it. For example, the 3-qubit operator $I_1 Y_2 Z_3$ has weight two.

We list some useful properties of Pauli operators which we make use in this work:

1. Pauli operators are unitary operators, i.e., they square to the identity:

$$X^2 = Y^2 = Z^2 = I \tag{C-3}$$

2. Any Pauli operator may be written as a product of the other two by the equation

$$\sigma_j \sigma_k = i\epsilon_{jkl}\sigma_l + \delta_{jk} I, \tag{C-4}$$

where ϵ_{jkl} and δ_{jk} are respectively the Levi-Civita and the Kronecker delta symbols.

3. By property 2, we can express any Pauli operator as $i^p X^{u_x} Z^{u_z}$, with $p = 0, 1, 2, 3$. By property 1, the entries of the vector (u_x, u_z) only need to take the values of 0 and 1.

4. Given a pair of elements of $\mathcal{G}_n$, they either commute or anti-commute. For $\mathcal{G}_1$, equal operators trivially commute. As for different operators:

$$\sigma_j \sigma_k + \sigma_k \sigma_j = i\sigma_l - i\sigma_l = 0, \tag{C-5}$$

where property 2 was employed. Thus,

$$\sigma_i \sigma_j = -\sigma_j \sigma_i \tag{C-6}$$

if $i \neq j$. Two elements of $\mathcal{G}_n$ commute (anti-commute) if they intersect non-trivially on an even (odd) number of qubits. For example, $X_1 X_2$ and $Y_1 Z_2$ commute since

$$X_1 X_2 Y_1 Z_2 = (-1)^2 Y_1 X_1 Z_2 X_2 = Y_1 Z_2 X_1 X_2. \tag{C-7}$$

5. By property 4, Pauli operators conjugate to the negative of itself,

$$\sigma_i \sigma_j \sigma_i^\dagger = -\sigma_j, \tag{C-8}$$

if $i \neq j$ and $\sigma_i, \sigma_j \neq \pm I$.

D
Quantum circuits and important gates

Quantum circuits are a helpful way of visualizing quantum algorithms and are widely employed throughout this work. We briefly review quantum circuitry notation and the main gates utilized in the stabilizer formalism: the Pauli operators $\{X, Y, Z\}$ and the Clifford gates $\{H, \mathrm{CNOT}, \mathrm{P}\}$. We also cite the *measurement* gate, which has a different quality from the others and plays an important role on stabilizer error correction codes. Figure D.3 summarizes all mentioned gates and quantum circuit's elements for quick reference. For more in-depth details about quantum circuits and gates, please refer to [10].

To start drawing a quantum circuit, one begins by placing the initial qubit states on the left side of the circuit (crescent index from top to bottom) and assign a horizontal wire, also known as a register or quantum channel, to each qubit. Operators, also called gates in analogy with classical computation, are then placed in order of application from left to right on the correspondent qubits they act. As a first example, Figure D.1 depicts the quantum circuit representation of applying operators U and G, in this order, to a quantum state $|\psi\rangle$.

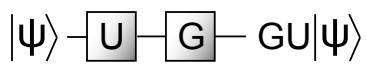

Figure D.1: A simple quantum circuit example.

We now review the principal gates employed in this dissertation. The X gate has the action of flipping the computational basis states:

$$X\left|0\right\rangle = \left|1\right\rangle, \; X\left|1\right\rangle = \left|0\right\rangle. \tag{D-1}$$

Hence, it is commonly called the NOT gate of quantum computation [10]. The diagonal basis $\{\left|+\right\rangle, \left|-\right\rangle\}$, defined as

$$\left|+\right\rangle \equiv \frac{\left|0\right\rangle + \left|1\right\rangle}{\sqrt{2}}, \; \left|-\right\rangle \equiv \frac{\left|0\right\rangle - \left|1\right\rangle}{\sqrt{2}}, \tag{D-2}$$

are the eigenvectors of X with eigenvalues ± 1. The Z gate has a similar action, however it flips the diagonal basis states:

$$Z\left|+\right\rangle = \left|-\right\rangle, \; Z\left|-\right\rangle = \left|+\right\rangle. \tag{D-3}$$

This comes from the fact that $\{|0\rangle, |1\rangle\}$ are eigenvectors of Z with eigenvalues ± 1. The Y gate is a mixture of the previous two gates since it is proportional to XZ. With the previous results, it immediately follows that its action upon a generic qubit state $\alpha|0\rangle + \beta|1\rangle$ is

$$Y(\alpha|0\rangle + \beta|1\rangle) = \alpha|1\rangle - \beta|0\rangle, \tag{D-4}$$

where the global phase was ignored. Its eigenvectors are $\left\{\frac{|0\rangle+i|1\rangle}{\sqrt{2}}, \frac{|0\rangle-i|1\rangle}{\sqrt{2}}\right\}$ with eigenvalues of ± 1.

The Hadamard gate H maps the states of the computational basis into the states of the diagonal basis:

$$H|0\rangle = |+\rangle, H|1\rangle = |-\rangle, H|+\rangle = |0\rangle, H|-\rangle = |1\rangle. \tag{D-5}$$

The phase gate P leaves the $|0\rangle$ state unaltered and gives a relative phase of i to the $|1\rangle$ state:

$$P(\alpha|0\rangle + \beta|1\rangle) = \alpha|0\rangle + i\beta|1\rangle. \tag{D-6}$$

Unlike all the other gates mentioned in this appendix, the phase gate does not square to identity. Actually, $P^4 = I$. The CNOT gate, short for *controlled*-NOT, is a 2-qubit gate with two input qubits: a *control* and a *target* qubit. Its action depends on the state of the control qubit. If the control is set to $|0\rangle$, the target is left unaltered. If the control is set to $|1\rangle$, it is applied an X gate to the target. Figure D.2 shows the quantum circuit representation of the CNOT in a useful application with the H gate. The target is properly symbolized by a big cross inside a circle, and the control by a small black circle. Applying the definitions, it is straightforward to check that this circuit, before the measurement, generates the Bell state $|\Phi^+\rangle = (|00\rangle + |11\rangle)/\sqrt{2}$.

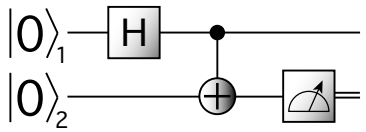

Figure D.2: This quantum circuit example produces the Bell state $|\Phi^+\rangle$ and then qubit 2 is measured on the computational basis.

Lastly, we define a measurement as a projection of a qubit into the computational base. For example, consider the circuit depicted in Figure D.2. The measurement gate is schematically represented by a needle pointer similar to a vintage voltmeter display. As aforementioned, the pre-measurement state for this circuit is the Bell state $(|00\rangle + |11\rangle)/\sqrt{2}$. By Born's rule, measurement leads the state to collapse into either $|0\rangle$ or $|1\rangle$ with equal probability of 0.5. Often one is not interested in the post-measurement state, only in the outcome,

which is a classical bit. Therefore is common to draw a double line coming out of a measurement gate, symbolizing a classical bit being transmitted.

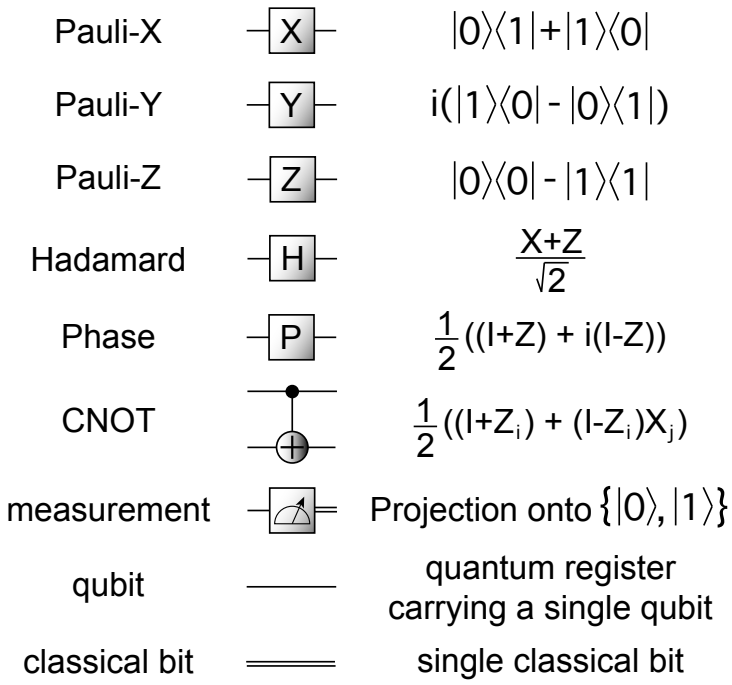

Figure D.3: Quick reference for quantum gates and quantum circuit's basic elements. The third column shows useful representation of gates in terms of the computational base or Pauli operators.

E
Summary and important properties of stabilizers

We give a summary of stabilizer states and list some important properties that are used throughout this work.

1. U stabilizes $|\psi\rangle$ iff $U|\psi\rangle = +|\psi\rangle$;

2. A n-qubit state $|\psi\rangle$ has 2^n stabilizers which form an Abelian group $\mathcal{S}(|\psi\rangle)$ [36];

3. $\mathcal{S}(|\psi\rangle) = \mathcal{S}(|\phi\rangle)$ iff $|\psi\rangle = |\phi\rangle$ [37];

4. The elements of $\mathcal{S}(|\psi\rangle)$ are elements of the general Pauli group $\mathcal{G}_n$ [36];

5. $\mathcal{S}(|\psi\rangle)$ can be represented by a not unique set of n generators $\langle\mathcal{S}(|\psi\rangle)\rangle$ [36];

6. The number of unique n-qubit stabilizer states is given by [37]

$$N(n) = 2^n \prod_{k=0}^{n-1} \left(2^{n-k} + 1\right). \tag{E-1}$$

7. If $+U|\psi\rangle = |\psi\rangle$ and $-U|\phi\rangle = |\phi\rangle$, then $\langle\psi|\phi\rangle = 0$. *Proof*:

$$\langle\phi|\psi\rangle = -\langle\phi|U^\dagger U|\psi\rangle = -\langle\phi|\psi\rangle, \tag{E-2}$$

   where $U^\dagger U = I$ since $U \in \mathcal{G}_n$. $\langle\phi|\psi\rangle = -\langle\phi|\psi\rangle \rightarrow \langle\psi|\phi\rangle = 0$;

8. If G stabilizes $|\psi\rangle$ and an operator U is applied to the state, $UGU^\dagger$ stabilizes the evolved state $U|\psi\rangle$;

9. Pauli letters maps to single products of Pauli letters under conjugation with Clifford gates (H,CNOT,P);

10. Stabilizer states are unbiased. An arbitrary state $|\psi\rangle$ with decomposition $\sum_{i=1}^{n} \lambda_i |c_i\rangle_i$, with $c \in \{0,1\}$, is said to be unbiased if for all $\lambda_i \neq 0$ and $\lambda_j \neq 0$, $|\lambda_i|^2 = |\lambda_j|^2$. Using Table 2.1, it is easy to convince yourself that an arbitrary Clifford circuit applied to an n-qubit initial stabilizer state on the computational base can only produce an unbiased state;

11. Consider a stabilizer state $|\psi\rangle$ with stabilizer group $\mathcal{S}(|\psi\rangle) = \{s\}$. The projector $P_{|\psi\rangle} \equiv |\psi\rangle\langle\psi|$ can be written as

$$P_{|\psi\rangle} = \frac{1}{|\mathcal{S}|}\sum_{s\in\mathcal{S}} s. \tag{E-3}$$

Proof: First we show that $P^2_{|\psi\rangle} = P_{|\psi\rangle}$, i.e., $P_{|\psi\rangle}$ is a projector since it squares to itself:

$$\begin{aligned} P^2_{|\psi\rangle} &= \frac{1}{|\mathcal{S}|^2}\sum_{s,h\in S} sh \\ &= \frac{1}{|\mathcal{S}|^2}\sum_{s\in\mathcal{S}}\left(s\sum_{h\in S} h\right) \\ &= \frac{1}{|\mathcal{S}|^2}\sum_{s\in\mathcal{S}}\left(\sum_{h\in\mathcal{S}} h\right) \\ &= \frac{1}{|\mathcal{S}|^2}|S|\sum_{h\in\mathcal{S}} h = P_{|\psi\rangle}. \end{aligned} \tag{E-4}$$

The third equality follows from $\{s\}$ being a group — multiplying a group by one of its elements maps each element to another distinct one [107] (it is a transposition). Apply $P_{|\psi\rangle}$ to an arbitrary state $|\phi\rangle$. Due to the same reasoning, it follows that $P_{|\psi\rangle}|\phi\rangle$ is stabilized by $\{s\}$:

$$sP_{|\psi\rangle}|\phi\rangle = \frac{1}{|\mathcal{S}|}s\sum_{h\in\mathcal{S}} h|\phi\rangle = \frac{1}{|\mathcal{S}|}\sum_{h\in\mathcal{S}} h|\phi\rangle = P_{|\psi\rangle}|\phi\rangle. \tag{E-5}$$

12. Any element $g \in \mathcal{G}_n$ can be expressed as a set $\{\vec{u}, p\}$, where $\vec{u}$ is a n-bit vector and $p = 0, 1, 2, 3$;

13. $2n^2 + n$ bits are sufficient to describe a stabilizer state.