UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

TIAGO MENDONÇA LUCENA DE VERAS

Circuit-based quantum random access memory for sparse quantum state preparation.

Recife

2021

TIAGO MENDONÇA LUCENA DE VERAS

Circuit-based quantum random access memory for sparse quantum state preparation.

Thesis presented to the Postgraduate Program in Computer Science from the Universidade Federal de Pernambuco, as a partial requirement for obtaining the title of Doctor of Computer Science.

Concentration area:Computation Theory.

Advisor: Prof. Dr. Ruy José Guerra Barretto de Queiroz
Co-advisor: Prof. Dr. Adenilton José da Silva.

Recife
2021

Catalogação na fonte
Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

V476c Veras, Tiago Mendonça Lucena de
Circuit-based quantum random access memory for sparse quantum state preparation / Tiago Mendonça Lucena de Veras. – 2021.
97 f.: il., fig., tab.

Orientador: Adenilton José da Silva.
Tese (Doutorado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2021.
Inclui referências e apêndice.

1. Teoria da computação. 2. Computação quântica. I. Silva, Adenilton José da (orientador). II. Título.

004 CDD (23. ed.) UFPE - CCEN 2021 – 161

TIAGO MENDONÇA LUCENA DE VERAS

“Circuit-based quantum random access memory for sparse quantum state preparation.”

Thesis presented to the Postgraduate Program in Computer Science from the Universidade Federal de Pernambuco, as a partial requirement for obtaining the title of Doctor of Computer Science.

Concentration area: Computation Theory

Aprovado em: 13/09/2021.

_____
Orientador: Prof. Dr. Ruy José Guerra Barreto de Queiroz

BANCA EXAMINADORA

_____
Prof. Dr. Fernando Maciano de Paula Neto
Centro de Informática / UFPE

_____
Prof. Dr. Antônio Murilo Santos Macedo
Departamento de Física / UFPE

_____
Prof. Dr. Franklin Marquezino
Departamento de Engenharia de Sistemas e Computação / UFRJ

_____
Prof. Dr. Leon Denis da Silva
Departamento de Matemática / UFRPE

_____
Prof. Dr. Tiago Alessandro Espínola Ferreira
Departamento de Informática e Estatística / UFRPE

I dedicate to everyone who will contribute to my work.

ACKNOWLEDGEMENTS

To the support, patience and company of my fiancée Agata Ferreira, my father Pedro Veras, my mother Ozani Veras, my brother Bruno Veras and all my family members.

To the support and company of my great life friends Marcela Dourado, Eduardo Figueira, Luciana Lira, Fredson Gonçalves, Edionara Celedonio, Cândido Lobo.

To my friends from work and day Leon Denis, José Deibsom, Eberson Ferreira, Maité Kulesza for the support and motivation.

To my advisor Ruy Queiroz for all his patience, availability and support in this professional stage.

To my co-advisor Adenilton Silva, for the immeasurable support in my training, for the patience to lead me in new fields of research that built this dissertation.

To my graduate friends Arthur Ramos, Jerônimo Júnior, Maigan Alcântara.

To the Department of Mathematics at UFRPE, where I am currently a professor, for all the support given during the completion of my doctorate.

To Cin-UFPE for a whole basic structure.

"I wll hit the bottom hit the bottom and escape Escape" (RADIOHEAD, 2003)

ABSTRACT

In order to use a quantum device to assess a classical dataset $\mathcal{D}$, we need to represent the set $\mathcal{D}$ in a quantum state. Applying a quantum algorithm, that is a quantum state preparation algorithm, to convert classical data into quantum data would be the common method. Loading classical data into a quantum device is required in many current applications. Efficiently preparing a quantum state to be used as the initial state of a quantum algorithm is an essential step in developing efficient quantum algorithms, since many algorithms need to reload the initial state several times during their execution. The cost to initialize a quantum state can compromise the algorithm efficiency if the process of quantum states preparation is not efficient. The topic of quantum states preparation in quantum computing has been the focus of much attention. In this scope, preparing sparse quantum states is a more specific problem that remains open since many quantum algorithms also require sparse initialization. This dissertation presents the results of an investigation on sparse quantum states preparation with the development of three algorithms, with highlight to the preparation of sparse quantum states, the main contribution of this dissertation. From a classical input dataset with M patterns formed by pairs composed of a complex number and a binary pattern with n bits, this algorithm can prepare a quantum state with n qubits and continuous amplitudes. The cost of its steps is $\mathcal{O}(nM)$, classical cost of $\mathcal{O}(M \log M + nM)$ and requires a lower CNOT number than the main quantum state preparation algorithms currently known. The preparation of a quantum state with 2^s non-zero amplitudes reveals the need of fewer CNOT gates in $n \gg 1$ relation to the main known state preparation algorithms, with even more favorable results with s higher and less $1s$ in the binary string.

Keywords: Quantum computation; Quantum random acess memory; Quantum state initialization; Quantum states preparation; Sparses quantum states.

RESUMO

Com objetivo de usar um dispositivo quântico para avaliar um conjunto de dados clássicos $\mathcal{D}$, precisamos representar o conjunto $\mathcal{D}$ em um estado quântico. Aplicar um algoritmo quântico, que é um algoritmo de preparação de estados quânticos, para converter dados clássicos em dados quânticos seria o método comum. Carregar dados clássicos em um dispositivo quântico é necessário em muitas aplicações atuais. A preparação eficiente de um estado quântico para ser utilizado como o estado inicial de um algoritmo quântico é uma etapa essencial no desenvolvimento de algoritmos quânticos eficientes, uma vez que muitos algoritmos precisam recarregar o estado inicial várias vezes durante sua execução. O custo para inicializar um estado quântico pode comprometer a eficiência do algoritmo se o processo de preparação dos estados quânticos não for eficiente. O tópico da preparação de estados quânticos na computação quântica tem sido o foco de muita atenção. Nesse escopo, a preparação de estados quânticos esparsos é um problema mais específico que permanece em aberto, uma vez que muitos algoritmos quânticos também requerem inicialização esparsa. Esta tese apresenta os resultados de uma investigação sobre a preparação de estados quânticos esparsos com o desenvolvimento de três algoritmos, com destaque para a preparação de estados quânticos esparsos, principal contribuição desta tese. A partir de um conjunto de dados de entrada clássico com M padrões, formados por pares compostos por um número complexo e um padrão binário com n bits, este algoritmo pode preparar um estado quântico com n qubits e amplitudes contínuas. O custo de passos é $\mathcal{O}(nM)$, o custo clássico é de $\mathcal{O}(M \log M + nM)$ e requer um número de CNOT menor do que os principais algoritmos de preparação de estado quântico conhecidos atualmente. Na preparação de um estado quântico com 2^s amplitudes diferentes de zero, revela a necessidade de menos portas CNOT quando $n \gg 1$ em relação aos principais algoritmos de preparação de estado conhecidos, com resultados ainda mais favoráveis com s maior e menor $1s$ na string binária.

Palavras-chaves: Computação quântica; Memória de acesso aleatório quântica; Inicialização de estados quânticos; Preparação de estados quânticos; Estados quânticos esparsos.

CONTENTS

1 INTRODUCTION

Features of quantum computing such as state superposition and quantum parallelism (NIELSEN; CHUANG, 2002) allow to obtain an acceleration in the ability to perform information processing, as well as specific quantum computing algorithms to be more efficient than classical algorithms. The first mechanical model of a quantum computer was described by Benioff (1980). In this study, Benioff presents computation models that operate under the laws of quantum mechanics, describing what would be a quantum version of the Turing machine and how it should work. The work published by Feynman (1982) can be considered a milestone of quantum computing. Feynman concludes that a classical computer cannot simulate efficiently a quantum system since simulating the results of quantum mechanics on a classical universal device would require an exponential amount of memory.

Although Deutsch (1985) proposed a quantum version of the classical Turing machine, using the quantum Turing machine is not the standard when building algorithms or performing implementations. Thus, Deutsch e Jozsa (1992) proposes an equivalent model, the quantum circuit model, which has been used more often.

The factorization algorithm proposed by Shor (1997) has a polynomial-time cost to factor an integer N into the product of primes. In contrast, the best-known classical algorithm has an exponential time cost. The quantum search algorithm proposed by Grover (1997) has a cost $\mathcal{O}(\sqrt{N})$ queries to find a value in an unsorted database, while classical algorithms have at least a time cost of $\mathcal{O}(N)$. Shor's and Grover's algorithms are examples of more efficient quantum algorithms than the known classical algorithms.

Quantum gates as CNOT that act simultaneously on two qubits are noisier and make the results more susceptible to errors than elementary quantum gates that work on a single qubit (BASSI; DECKERT, 2008),

Quantum gates that act on more than two qubits are yet to receive implementations on current quantum hardware. As showed by Barenco et al. (1995), all quantum programming can be designed using elementary quantum gates and CNOT gates.

The number of CNOT gates needed to implement multi-controlled gates that act on three or more qubits increases linearly with respect to the number of control qubits. Since multi-controlled gates cause an even greater noise when executing an algorithm, a lower number of CNOT gates in the algorithm implementation can allow to implement the algorithm in NISQ devices.

Currently, the ability to load classical data into a quantum device is an essential task when studying the efficiency of quantum algorithms (BIAMONTE et al., 2016). This procedure is a subroutine of many quantum algorithms and is called quantum state preparation.

The quantum state preparation has been a researched subject for more than two

decades, as shown in (VENTURA; MARTINEZ, 1970) (GROVER, 2000) (KAYE; MOSCA, 2001)(LONG; SUN, 2001), which are among the first papers addressing the theme of state preparation. There are numerous algorithms that require the preparation of an initial quantum state, for instance, in the solution of linear systems (HARROW; HASSIDIM; LLOYD, 2009) (BERRY et al., 2017), discrete time quantum walk (CHILDS, 2008) (LOW; CHUANG, 2016), solving optimization problems (SOMMA et al., 2008), quantum machine learning (BIAMONTE et al., 2016) (KIEFEROVA; WIEBE, 2017), and simulation of physical systems (LLOYD, 1996) (ZALKA, 1998)

The central topic of this dissertation is introduce the research results developed in the scope of the quantum states preparation. Conducting a quantum state to the desired target state is a difficult task (GIROLAMI, 2019) and requires thorough consideration in the best possible way. According to the third postulate of quantum mechanics, any superposed quantum state collapses to a basis state after taking a measurement. Furthermore, the state cannot be copied according to the non-cloning theorem (WOOTTERS; ZUREK, 1982). Consequently, many algorithms need to reload the initial state several times during its execution, increasing the processing time and data exposure, which can compromise the results obtained from the action of noisy quantum operations (PRESKILL, 2018) and the decoherence of quantum information (HUGHES et al., 1996).

Measuring the number of CNOT gates needed to prepare a quantum state is a valuable way to measure the efficiency of a quantum algorithm, since CNOT gates generate the most noise. We divide the state preparation algorithms into two classes.

The first class contains the algorithms proposed in (TRUGENBERGER, 2001) (PARK; PETRUCCIONE; RHEE, 2019), whose cost regarding the number of CNOT gates required depends on the input number M and the number of qubits n, with linear classical cost to build circuits according to the number of qubits and the input patterns. The second class contains the algorithms proposed in (SHENDE; BULLOCK; MARKOV, 2006)(PLESCH; BRUKNER, 2011) (MÖTTÖNEN et al., 2005), whose cost regarding the number of CNOT gates required depends exclusively on the number of qubits n. This class has an exponential classical cost to build circuits according to the number of qubits.

A probabilistic quantum state preparation algorithm is proposed in (PARK; PETRUCCIONE; RHEE, 2019), called FF-QRAM. The FF-QRAM algorithm is a QRAM built on the quantum circuit model consisting of $\mathcal{O}(n)$ qubits and has the flexibility to use modern quantum computing techniques. From a classical input dataset $\mathcal{D}$ containing M input patterns of type (x_k, p_k), where each x_k is a complex number and p_k is a binary pattern with n bits, the FF-QRAM prepares a quantum state $|\psi\rangle$ with n qubits and continuous amplitudes. To load or update input patterns M into a quantum state $|\psi\rangle$, FF-QRAM demands a classical cost $\mathcal{O}(nM)$ to build the circuit and requires C times $\mathcal{O}(nM)$ steps performed by the Flip-flop quantum register QRAM, where C denotes the necessary post-selection cost performed by the algorithm to select the correct result.

Post-selection is the space probability conditioning for an event to occur; thus, it can discard the results without the occurrence of a given event. In quantum computing, post-selection is the power to discard an execution of the quantum algorithm that achieved a success probability in the post-measurement $P(|1\rangle \simeq 0)$.

When FF-QRAM was first introduced, the extra cost for post-selection was neglected. This post-selection can compromise algorithm efficiency since this cost is associated differently with values M and n. Thus, in current quantum computing, it is still a critical problem to build an efficient quantum state preparation algorithm capable of receiving classical data as input and outputting a prepared quantum state, without requiring post-selection.

The first part of the work Trugenberger (2001) proposes a deterministic quantum state preparation algorithm called PQM. The PQM is a QRAM built on the quantum circuit model consisting of $\mathcal{O}(n)$ qubits and receives as input pattern M classical binary patterns of n bits denoted by $p_k \in \{0,1\}^n$, and provides as outputs a quantum state prepared in superposition with uniform amplitudes. The PQM has a cost of $\mathcal{O}(nM)$ steps to store M input pattern, without requiring post-selection, requiring $20n - 2$ CNOT gates to load each binary string of n bits into a quantum state with n qubits and classical cost $\mathcal{O}(nM)$ to build the circuit.

FF-QRAM has the advantage of loading continuous amplitudes and requiring fewer CNOT gates to load a pattern into memory in relation to PQM. Considering its probabilistic nature, post-selection is the main disadvantage of FF-QRAM and is likely to cause severe difficulties in the sparse quantum states preparation. In contrast, PQM has the advantages of not requiring post-selection and having a computational cost for the number of steps depending on the number of input patterns and the number of qubits. Loading uniform amplitudes and requiring a greater number of CNOT gates to prepare a state in memory comparing with FF-QRAM is a disadvantage of PQM. Both algorithms have the advantage of having a classical linear cost when constructing the circuit.

By observing the disadvantages and advantages of FF-QRAM and PQM algorithms and other state preparation algorithms, we learn that the construction of a sparse quantum states preparation algorithm that does not require post-selection involving computational cost $\mathcal{O}(nM)$ steps, in which CNOT cost depends on M and n, has a non-exponential classical cost for the circuit. This is an open problem in the quantum state preparation topic.

1.1 OBJECTIVES

This open problem drove our main goal and specific goals regarding the quantum state preparation from a $\mathcal{D}$ with M patterns of non-null input and n qubits.

To build a deterministic algorithm for sparse quantum states preparation with computational cost $\mathcal{O}(nM)$, polynomial classical cost, and CNOT cost below the main quantum

state preparation algorithms currently known. More specifically, the proposed algorithm is more efficient than both the FF-QRAM and the PQM algorithms for sparse states. It uses a smaller number of CNOT gates than algorithms with exponential cost regarding the number of qubits. In the double sparse quantum state case, it requires a smaller number of CNOT operators than the sparse state preparation algorithm recently proposed by Malvetti, Iten e Colbeck (2021) for $n \gg 1$.

The proposed algorithm is based on FF-QRAM and PQM algorithms. This work has the following specific objectives:

- To improve the post-selection probability of FF-QRAM, which is equivalent to changing the computational cost from $\mathcal{O}(CnM)$ to $\mathcal{O}(C'nM)$ steps, where $C' < C$.
- To have a computational cost of $\mathcal{O}(nM)$ steps without requiring post-selection, capable of carrying continuous amplitudes and with polynomial classical cost.
- A deterministic algorithm that loads continuous amplitudes to reduce the CNOT cost in relation to the main state preparation algorithms currently known, with a computational cost of $\mathcal{O}(nM)$ steps and polynomial classical cost.

1.2 WORKS DEVELOPED DURING THE DOCTORAL RESEARCH.

The works developed during the doctoral research are listed below.

- **DE VERAS, TIAGO M.L.** ; ARAUJO, ISMAEL ; PARK, D. K. ; DA SILVA, ADENILTON J. . Circuit-based quantum random access memory for classical data with continuous amplitudes. IEEE TRANSACTIONS ON COMPUTERS, v. x, p. x, 2020.

  Several quantum computing applications require to load data in a quantum device, and without an efficient loading procedure, the cost to initialize the algorithms can prevail in the overall computational cost. A circuit-based quantum random access memory named FF-QRAM can load M n-bit patterns at computational cost $O(CMn)$ to load continuous data where C depends on the data distribution. Our study proposes a strategy to load continuous data without post-selection at computational cost $O(Mn)$. The proposed method is based on the probabilistic quantum memory, a strategy to load binary data in quantum devices, as well as the FF-QRAM using standard quantum gates, suitable for noisy intermediate-scale quantum computers.

- **TIAGO M. L. DE VERAS**, LEON D. DA SILVA, ADENILTON J. DA SILVA Double sparse quantum state preparation. arXiv preprint 2108.13527.(2021).

  Initializing classical data in a quantum device is an essential step in many quantum algorithms. As a consequence of measurement and noisy operations, some algorithms

need to reinitialize the prepared state several times during its execution. If the quantum state preparation is not efficient, its cost can prevail in the computational cost of an algorithm. This study proposes a quantum state preparation algorithm, called CVO-QRAM, at computational cost $\mathcal{O}(kM)$, where M is the number of nonzero probability amplitudes and k is the maximum number of bits with value 1 in one of the patterns to be stored. The proposed algorithm can represent an alternative to create sparse states in future NISQ devices.

- SOUSA, RODRIGO S. ; DOS SANTOS, PRISCILA G.M. ; **VERAS, TIAGO M.L. ; DE OLIVEIRA, WILSON R. ; DA SILVA, ADENILTON J.** . Parametric Probabilistic Quantum Memory. NEUROCOMPUTING, v. 416, p. 360-369, 2020.

  Probabilistic Quantum Memory (PQM) is a data structure that computes the distance from a binary input to all binary patterns stored in superposition on the memory. This data structure allows the development of heuristics to speed up artificial neural networks architecture selection. In this work, we propose an improved parametric version of the PQM to perform pattern classification, in addition to presenting a PQM quantum circuit suitable for Noisy Intermediate Scale Quantum (NISQ) computers. We introduce a classical assessment of a parametric PQM network classifier on public benchmark datasets. We also perform experiments to verify the viability of PQM on a 5-qubit quantum computer.

- RAMOS, A.F. ; QUEIROZ, R. J. G. B. ; OLIVEIRA, A. G. ; **DE VERAS, TIAGO M.L.** . Explicit Computational Paths. SOUTH AMERICAN JOURNAL OF LOGIC, v. 4, p. 441/484-484, 2019.

  Addressing equality as a type of theory engenders an interesting type-theoretic structure known as 'identity type'. The idea is that given terms a, b of a type A, it is possible to form the type $Id_A(a, b)$, whose elements prove that a and b are equal elements of type A. A term of this type, $p : Id_A(a, b)$, makes up for the grounds (or proof) that establishes that a is indeed equal to b. On that basis, a proof of equality can be regarded as a sequence of substitutions and rewrites, also known as computational path. One interesting fact is that it is possible to rewrite computational paths using a set of reduction rules arising from an analysis of redundancies in paths. These rules were mapped by De Oliveira in 1994 in a term rewrite system known as LND_{EQ-}TRS. Our study uses computational paths and such term rewrite system to develop the main foundations of homotopy type theory, i.e., we develop the lemmas and theorems connected to the main types of this theory, types such as products, coproducts, identity type, transport, among many others. We also demonstrate that it is possible to construct path spaces directly through computational paths by using our path-based approach to construct two important structures of homotopy type

theory: the path-space of natural numbers and the construction and calculation of the fundamental group of the circle.

1.3 WORK STRUCTURE

The remainder of this work is divided into four chapters:

Chapter 2 defines qubits and quantum gates, in addition to describing the main quantum gates to be used. It also introduces the quantum RAM applied to initialize quantum states and the decomposition of multiple qubit gates into one qubit gate and CNOT gates to be adopted for multi-controlled quantum gates of type C^nU.

Chapter 3 describes the main state preparation algorithms related to our research.

Chapter 4 presents the three quantum state preparation algorithms developed throughout this dissertation. Section 4.1 contains the first two algorithms, called Pre-processed FFP-QRAM and CV-QRAM, while section 4.2 contains the CVO-QRAM.

Chapter 5 introduces our conclusions, main contributions, and further research. Finally, the Appendix contains the paper "Parametric Probabilistic Quantum Memory", developed and published in the course of the dissertation.

2 METHODS

Quantum mechanics is a mathematical and conceptual framework for the development of physical theories. It is composed of four basic postulates that establish the connection between the physical world and the formalism of quantum mechanics.

According to **Postulate 1** of quantum mechanics, any isolated physical system is associated with a Hilbert space (complex vector space with the inner product). This system is completely described by its state vector, which is a unit vector in state space.

A bit is a unit of classical information whose basis states are 0 and 1. A quantum bit or qubit is the simplest quantum system and represents the unit of quantum information whose basis states are $|0\rangle$ and $|1\rangle$. Unlike the classical bit, which can only be in 0 or 1, qubits can assume infinite states, which derives from the following definition:

Definition 1. *(Nakahara (2008)) A quantum state*

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{2.1}$$

is a unit of complex two-dimensional vector, with $\alpha, \beta \in \mathbb{C}^2$*, satisfying* $|\alpha|^2 + |\beta|^2 = 1$*, where the quantum state base can be represented by the following matrices:*

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \text{ and } |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

According to **Postulate 2** of quantum mechanics, the evolution of a closed quantum system is described by a linear unitary transformation U. That is, U maps a quantum state $|\psi_1\rangle$ in an instant t_1 into a quantum state $|\psi_2\rangle$ in an instant t_2. Using this operator, it depends only on time t. Thus, we can define quantum operations as follows.

Definition 2. *A quantum operator* U *is a linear unitary operator defined over the complex vector space, which maps a quantum state* $|\psi_1\rangle$ *into another quantum state* $|\psi_2\rangle$*, that is*

$$\begin{aligned} U: \quad \mathbb{C}^n &\longrightarrow \mathbb{C}^n \\ |\psi_1\rangle &\longrightarrow |\psi_2\rangle \end{aligned}$$

where $U|\psi_1\rangle = |\psi_2\rangle =$*,* $U^\dagger U = \mathbb{I}$ *with* $U^\dagger = U^{-1}$*.*

Given a vector v, the quantum operators that rotate this vector at an angle θ, around the axis x, y, z, are given respectively by

$$R_x(\theta) = \begin{bmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix}, \quad R_y(\theta) = \begin{bmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix}, \quad R_z(\theta) = \begin{bmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{bmatrix}.$$

From these rotation operators, Eq. 2.3 can be rewritten as follows.

$$\begin{aligned} R_z(-\phi)|\psi\rangle &= e^{it/2}\Big[\cos\frac{\theta}{2}|0\rangle + \sin\frac{\theta}{2}|1\rangle\Big] \\ R_y(-\theta)R_z(-\phi)|\psi\rangle &= e^{it/2}|0\rangle \\ R_z(t)R_y(-\theta)R_z(-\phi)|\psi\rangle &= |0\rangle. \end{aligned}$$

We can conclude that any state $|\psi\rangle$ given by Eq. (2.1) can be prepared from $|0\rangle$, since:

$$|\psi\rangle = R_z^\dagger(-\phi)R_y^\dagger(-\theta)R_z^\dagger(t)|0\rangle, \tag{2.2}$$

and this result can be generalized to a quantum state with n qubits. The quantum state described by Eq. (2.2) can be rewritten as:

$$|\psi\rangle = e^{it/2}\Big[e^{-i\phi/2}\cos\frac{\theta}{2}|0\rangle + e^{i\phi/2}\sin\frac{\theta}{2}|1\rangle\Big], \tag{2.3}$$

where $\theta \in [0, \pi]$, $\phi \in [0, 2\pi)$ and $t \in [0, 2\pi)$.

There is a geometrical representation to the quantum state described in Eq. (2.2), known as the Bloch sphere (YANOFSKY; MANNUCCI; MANNUCCI, 2008).

Preparing an arbitrary quantum state $|\psi\rangle$ as initial input is an essential subroutine required in many quantum algorithms (LAROSE; COYLE, 2020), (KOTHARI, R., 2014), (BERRY et al., 2014), (SCHERER et al., 2017), (SCHULD; KILLORAN, 2018). In practice, these quantum algorithms demand efficient load or update input patterns, consisting of classical data, into quantum computers (BIAMONTE et al., 2016).

This is equivalent to the algorithm to receive a classical data set $data = \{x_0, x_1, x_2, \ldots, x_{M-1}\}$ where $x_k \in \mathbb{C}$ such that $\sum_{i=0}^{k-1} |x_i|^2 = 1$, providing as output a quantum state

$$|\psi\rangle = \sum_{k=0}^{M-1} x_k |p_k\rangle_n, \tag{2.4}$$

where $p_k \in \{0,1\}^n$ is a n binary pattern, and a quantum state with n qubits for each k $|p_k\rangle$.

According to **Postulate 3** of quantum mechanics, quantum measurements are described as a collection of $\{M_m\}$ measurement operators that act on the system state space, where the index m denotes the outputs from the measurement. If $|\psi\rangle$ is a quantum

system state before the measurement, then the probability of reaching m as a result is given by

$$P(m) = \langle\psi|M_m^\dagger M_m|\psi\rangle,$$

so, after measurement, status will be:

$$\frac{M_m|\psi\rangle}{\sqrt{\langle\psi|M_m^\dagger M_m|\psi\rangle}}.$$

Measurement operators must satisfy the completeness relationship $\sum_m M_m^\dagger M_m = I$, where the completeness relationship expresses that $\sum_m P(m) = 1$.

Considering that many algorithms need to execute data processing several times, their successful execution requires an efficient preparation of an initial quantum state. According to the **Postulate 3** of quantum mechanics, any quantum state that undergoes a measurement collapses to one of the basis states. Since many algorithms need to initialize the initial quantum state several times during execution, the problem of quantum state preparation is an indispensable step when constructing efficient algorithms.

Assuming that we wish to prepare a quantum state from M non-null input patterns with n qubits, and if t denotes the number of $1s$ in the binary string, then we establish the following definitions regarding quantum states:

Definition 3 (Dense quantum state)**.** *We say the quantum state is dense if the number of non-null input patterns is* $M = 2^n$.

Definition 4 (Sparse quantum state)**.** *We say the quantum state is sparse if the number of non-null input patterns is* $0 < M \ll 2^n$.

Definition 5 (Double sparse quantum state)**.** *The quantum state is double sparse if the number of non-null input patterns is* $0 < M \ll 2^n$ *and* $0 < t \ll n$*, that is, the desired quantum state has few non-null patterns compared to* 2^n *and few* $1s$ *in the binary string.*

The ability to store information in a memory array is a fundamental skill in computing devices (FEYNMAN, 2000). On classical computers, the component called random access memory (RAM) is currently the most flexible and widely used architecture for memory arrays (JAEGERS; BLALOCK, 2003).

Quantum random access memory (QRAM) is an essential component in the process of storing information when dealing with quantum computers (GIOVANNETTI; LLOYD; MACCONE, 2008). QRAM is the quantum version of RAM consisting of a RAM that functions to preserve quantum coherence (NIELSEN; CHUANG, 2002). QRAM uses qubits in the address and output records. Both RAM and QRAM are high-cost computational devices.

The initialization of quantum data can be achieved by utilizing a quantum random access memory (QRAM). Thus, this study (GIOVANNETTI; LLOYD; MACCONE, 2008) proposes a QRAM architecture called a bucket-brigade Quantum Ram (BB-QRAM) using $O(M)$ quantum hardware components, as well as $O(\log_2^2(M))$ time steps to return binary data stored in M memory cells in superposition. This procedure can be expressed as

$$\frac{1}{\sqrt{M}}\sum_{k=0}^{M-1}|k\rangle|0\rangle \xrightarrow{QRAM} \frac{1}{\sqrt{M}}\sum_{k=0}^{M-1}|k\rangle|p_k\rangle, \tag{2.5}$$

where $|p_k\rangle$ is the content in the k-th memory cell and $p_k \in \{0,1\}^n$ is a n binary pattern, and for each k $|p_k\rangle$ is a quantum state with n qubits. Unfortunately, it is difficult to build a quantum circuit based on this model keeping its advantages (ARUNACHALAM et al., 2015)(MATTEO; GHEORGHIU; MOSCA, 2020).

Since the operations are carried out using quantum gates that are applied on one qubit, it is recommended that the quantum state used as the initial state of a quantum algorithm is built by a QRAM based on the circuit model (DEUTSCH; JOZSA, 1992).

Fast QRAM is often influenced by access speeds, as shorter times produce less noise caused by data exposure during algorithm execution (PALER; OUMAROU; BASMADJIAN, 2020). Estimating the resources needed to correct errors in a quantum algorithm is a critical point in the preparation and development of algorithms used in quantum computers. Therefore, QRAMs must be built in a fast and resource-efficient way.

2.1 DECOMPOSITION OF CONTROLLED QUANTUM GATES

The computational complexity of an algorithm is a mathematical analysis of its performance during execution (CORMEN et al., 2009). The number of operations, execution time and memory usage are examples of variables that can quantify the complexity of an algorithm to perform its task (CORMEN et al., 2009). Elementary quantum gates are unitary transformations that act on one or two qubits, and the number of elementary gates can measure the complexity of a quantum circuit included (MÖTTÖNEN et al., 2005).

Recently, quantum algorithms of quantum state preparation have improved the efficiency of gate synthesis by implementing quantum gates that act simultaneously on three or more qubits using elementary quantum gates (MÖTTÖNEN et al., 2005)(SHENDE; BULLOCK; MARKOV, 2006)(VARTIAINEN; MÖTTÖNEN; SALOMAA, 2004).

In real quantum devices, it is difficult to implement quantum gates that act simultaneously on three or more qubits. Currently, this is possible by decomposing these multi-controlled gates into gates that act on one or two qubits (BARENCO et al., 1995). CNOT gates are currently the only 2-qubit gates implemented in some quantum computer hardware, whose implementation makes the results of the algorithms more susceptible to errors than gates that act on a single qubit (BASSI; DECKERT, 2008). Thus, controlled quantum gates that act simultaneously on three or more qubits cause more noise to the system

than controlled gates that act on two qubits since multi-controlled gates generate more CNOT gates.

Therefore, it is crucial to design circuits with the fewer CNOT gates as possible to avoid imperfections caused by noise. Thus, the cost of a quantum algorithm (complexity) can be estimated by counting CNOT gates, which is our proposal to measure the cost of the algorithms here proposed.

Among the multi-controlled gates that act on multiple qubits, we define below a gate of fundamental interest in our work:

Definition 6. *Suppose that* $|\psi\rangle = |x_0 x_1 \dots x_{n-1} y\rangle$ *is a quantum state with* $n+1$ *qubits and* U *is a unitary operator that act on a single qubit. Then, we define the controlled operation* $C^n U$ *by the equation:*

$$C^n U|\psi\rangle = |x_0 x_1 \dots x_{n-1}\rangle U^p |y\rangle, \tag{2.6}$$

where $p = x_0 x_1 \dots x_{n-1}$ *is a product of the bits. Note that* U *operator is applied only if* $p = 1$*. All* x_i *bits are designated as control bits, and the* y *bit is designated as target bit.*

This is equivalent to saying that the one operator is applied only when all x_i bits are simultaneously equal to one; otherwise, nothing is done.

In the particular case $n = 1$ and $|\psi\rangle = |xy\rangle$

$$CU|\psi\rangle = |x\rangle U^x |y\rangle$$

denotes a 1-controlled operation, where x is the control bit and y is the target bit.

We use a $C^n U$ gate decomposition in elementary gates that act on one or two qubits, based on, (BARENCO et al., 1995), (NIELSEN; CHUANG, 2002).

2.1.1 Decomposition method

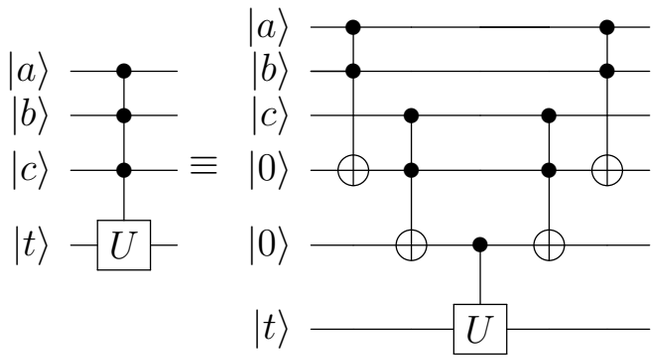
Our decomposition of $C^n U$ gates results from the combination of the decompositions of quantum gates, carried out in (NIELSEN; CHUANG, 2002) and (BARENCO et al., 1995).

According to (NIELSEN; CHUANG, 2002), the quantum gate $C^n U$ can be decomposed into $2(n-1)$ Toffoli gates and one single quantum gate CU using $n-1$ auxiliary qubits prepared in $|0\rangle$. For example, Figure 1 illustrates the decomposition of a $C^3 U$ gate as proposed in (NIELSEN; CHUANG, 2002).

After the decomposition proposed in (NIELSEN; CHUANG, 2002), all Toffoli gate generated are reversed, which enables to apply the decomposition proposed in (BARENCO et al., 1995), which uses 3 CNOT gates to implement one Toffoli gate, as shown in Figure 2.

In Figure 2, A denotes the quantum gate $R_y(\frac{\pi}{4})$, the symbol " $\simeq$ " indicates that the circuits are not identical, since when applying such implementation to the particular case

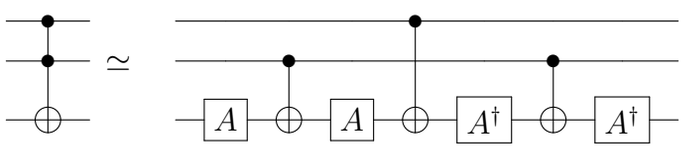
Figure 1 – Decomposition of a C^3U as proposed in the (NIELSEN; CHUANG, 2002)



Source: The author (2020)

$|101\rangle$ generates the sign of the switched phase. However, if applied to pairs, this Toffoli implementation establishes an equality.

Figure 2 – Decomposition of a Toffoli gate using 3 CNOT gates.



Source: The author (2020)

Therefore, the implementation of a C^nU gate, as mentioned above, requires $6n - 4$ CNOT gates.

3 QUANTUM STATE PREPARATION ALGORITHMS

This chapter introduces a brief description of some quantum state preparation algorithms, in addition to specifications on the cost of CNOT gates required to prepare the quantum state, since this parameter will be adopted to measure the algorithm efficiency. We will mention some works on quantum state preparation in more detail in the documents contained in the appendix.

3.1 DESCRIPTION OF WORK RELATED TO THE TOPIC.

This section describes some characteristics of each quantum state preparation algorithm chosen in our study, besides listing the negative points that were fundamental to establish our proposed algorithms.

3.1.1 Probabilistic Quantum Memories Algorithm - PQM Algorithm

The probabilistic quantum memory initialization algorithm was proposed in (TRUGENBERGER, 2001) and (VENTURA; MARTINEZ, 1970), which we will designate as PQM, a deterministic algorithm that receives as input M binary patterns $p_k \in \{0,1\}^n$ with n bits and is able to store them in a quantum state in superposition with uniform amplitudes as given below:

$$|\psi\rangle = \frac{1}{\sqrt{M}} \sum_{k=0}^{M-1} |p_k\rangle^{\otimes n}. \tag{3.1}$$

The PQM algorithm uses three quantum registers, totaling $2n+2$ qubits, including two auxiliary qubits. In order to store the M binary patterns with n bits, the algorithm has a computational cost (total operations) $O(nM)$ and uses $14n-2$ CNOT gates to load each pattern binary into the registry memory.

The PQM is a very specific quantum state preparation algorithm since the prepared state $|\psi\rangle$ has the same amplitude in all terms of the state. Preparing a quantum state with uniform amplitudes requires a large number of controlled operations in this process, representing the main disadvantages of the PQM algorithm.

3.1.2 Uniformly Controlled Rotations algortihm - UCR Algorithm

The algorithm proposed in (MÖTTÖNEN et al., 2005), called UCR, receives as quantum state $|\psi_a\rangle^{\otimes n}$ and uses a series of uniformly controlled rotations to prepare a desired quan-

tum state $|\psi_b\rangle^{\otimes n}$. The cost preparation of the UCR algorithm depends only on the number of n qubits of the state to be prepared. This procedure can be expressed as follows:

$$|\psi_a\rangle = \sum_{k=0}^{M-1} x_k |a_k\rangle^{\otimes n} \xrightarrow{UCR\ Algorithm} |\psi_b\rangle = \sum_{k=0}^{M-1} \beta_k |y_k\rangle^{\otimes n}, \tag{3.2}$$

where $M = 2^n$, x_k and $y_k \in \mathbb{C}$.

The algorithm applies a U sequence of uniformly controlled rotations in a quantum state $|\psi_a\rangle$ to obtain a desired quantum state $|\psi_b\rangle$ that is $U|\psi_a\rangle = |\psi_b\rangle$. As it is possible to obtain two arbitrary sequences of quantum operators X and Y where $X|\psi_a\rangle = |0\rangle = Y|\psi_b\rangle$ and setting $U = Y^\dagger X$, we have $Y^\dagger X|\psi_a\rangle = |\psi_b\rangle$. Thus, a given quantum state $|\psi_a\rangle$ allows to obtain a new desired quantum state $|\psi_a\rangle$ performing a sequence of quantum operations.

The UCR algorithm needs $2^{n+2} - 4n - 4$ CNOT gates to prepare a desired quantum state with n qubits. Additionally to such quantum cost, it involves a classical cost to generate the angles to be used in the $2^{n+2} - 5$ elementary rotations of a qubit. This algorithm may not be a good choice for preparing sparse quantum state as it depends exclusively on the number of qubits. The algorithm has the same cost to prepare both dense and sparse quantum states, which becomes a negative point due to the exponential costs. Algorithm 1 regarding the quantum state preparation proposed in (ARAÚJO et al., 2021) describes in detail the quantum state preparation process by means of the UCR.

3.1.3 Synthesis of Quantum Logic Algorithm - SQL Algorithm

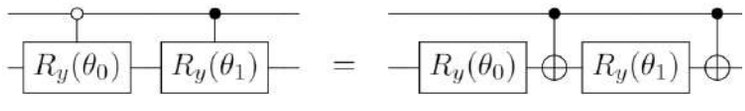
SQL is a quantum state preparation algorithm of synthesis of quantum logic proposed in (SHENDE; BULLOCK; MARKOV, 2006). In a first preparation step, the authors show that any quantum state with n qubits can be prepared from the basis states $|0\rangle^{\otimes n}$ or $|1\rangle^{\otimes n}$ through a sequence of operations using quantum gates $R_k(\theta)$, where $k = y, z$.

For $n = 1$, it is possible to obtain angles t, θ, ϕ, such that an arbitrary quantum state of a single qubit can be obtained from $|0\rangle$ as follows:

$$|\psi\rangle = R_z^\dagger(-\phi) R_y^\dagger(-\theta) R_z^\dagger(t) |0\rangle.$$

Subsequently, the authors establish a quantum version for classical conditional *if-then-else* where *true* and *false* can be replaced with open control and closed control, respectively.This is represented by the following quantum circuit in figure 3:

Figure 3 – The left side of the equality shows the quantum conditional for *if-then-else*, whereas the right side presents an identity that will serve us well.



Source: The author (2020)

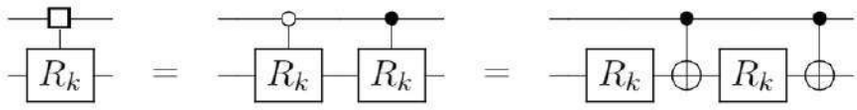
This quantum conditional can be represented by the U operator $U = U_0 \oplus U_1$ in the following matrix form, which can be written in the form of a diagonal matrix, given by:

$$U = \begin{pmatrix} U_0 & \\ & U_1 \end{pmatrix}$$

When applied to an arbitrary quantum state $|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$, the U operator provides as output the application of the operator U_0 or U_1 according to the selected qubit value. Thus, they define that a U gate is a quantum multiplexor with a s qubit selector. Each selector in the quantum circuit is denoted by the symbol $\square$, which represents the possibilities of the bit assuming the value 0 or 1.

A singly-multiplexed R_k for $k = y$ or $k = z$ can be demultiplexed according to the equality given in the figure 4

Figure 4 – Demultiplexing a quantum multiplexer with a single selector $\square$.



Source: The author (2020)

A decomposition for a quantum multiplexer with n selectors can be created recursively in a new circuit that uses only gates that act on a maximum of two qubits, such gates are operators rotation R_k, with $k = x, y$ and CNOT gates. The cost of the SQL algorithm depends only on the number of qubits, and to prepare a quantum state with n qubits the algorithm needs at most $2^{n+1} - 2n$ CNOT gates.

3.1.4 Universal Gate Decomposition Algorithm - UGD Algorithm

Consider that $\mathcal{H} = \mathcal{H}_A \otimes \mathcal{H}_B$ is the Hilbert space in a bipartite system AB. Then, a state $|\psi\rangle_{AB} \in \mathcal{H}$ admits the Schmidt decomposition

$$|\psi\rangle_{AB} = \sum_{k=0}^{M-1} x_k |\varphi_k\rangle_A \otimes |\phi_k\rangle_B, \tag{3.3}$$

where x_k are positive real coefficients called Schmidt coefficients, with $\sum_{k=0}^{M-1} |x_k|^2 = 1$; $\{|\varphi_k\rangle\}$ is a base of $\mathcal{H}_A$ and $\{|\phi_k\rangle\}$ is a base of $\mathcal{H}_B$.

We desigante as UGD the quantum state preparation algorithm proposed in (PLESCH; BRUKNER, 2011) using decompositions of universal quantum gates and a library of gates that act on two qubits during the quantum state preparation. As $|0\rangle^{\otimes n} \in \mathcal{H}$ is a separable state, it can be written as the tensor product according to the value of n.

- If $n = 2k$ (even), the quantum $|0\rangle^{\otimes n}$ is separated into two quantum states of size $\frac{n}{2}$, given by $|0\rangle^{\otimes n} = |0\rangle^{\otimes k} \otimes |0\rangle^{\otimes k}$.

- If $n = 2k + 1$ (odd), the quantum $|0\rangle^{\otimes n}$ is separated into two quantum states, the first state has $\frac{n-1}{2}$ qubits and the second $\frac{n+1}{2}$ qubits given by $|0\rangle^{\otimes n} = |0\rangle^{\otimes k} \otimes |0\rangle^{\otimes k+1}$.

The UGD algorithm uses four phases to prepare the quantum state with n qubits and $M = 2^n$ terms. If $n = 2k$, the algorithm is initialized at $|0\rangle^{\otimes 2k} = |0\rangle^{\otimes k}|0\rangle^{\otimes k}$. In the first phase, a quantum state will be prepared $|0\rangle^{\otimes k}$ leftmost with 2^k terms in the computational base of size k whose amplitudes are the Schmidt coefficients

$$|0\rangle^{\otimes k} \xrightarrow{\text{Schmidt coefficients}} \sum_{i=0}^{2^k-1} x_i |i\rangle^{\otimes k}.$$

This provides the quantum state given by

$$\left(\sum_{i=0}^{2^k-1} x_i |i\rangle^{\otimes k} \right) |0\rangle^{\otimes k}.$$

In the next steps, sequences of CNOT gates and unitary operations are used to obtain the quantum state desired by the UGD algorithm, so that from the beginning to the end of the process, we have

$$|0\rangle^{\otimes 2k} \xrightarrow{UGD\ -\ Phase\ 1} \left(\sum_{i=0}^{2^k-1} x_i |i\rangle^{\otimes k} \right) |0\rangle^{\otimes k}.$$

The second phase applies a set of CNOT gates between the leftmost k qubits and the rightmost k qubits. Each CNOT has control over qubit j and targets in qubit $j+k$, where j varies from $1, \ldots, k$. With this, the states of k qubits of each terms are equalized, thus generating the following quantum state:

$$\left(\sum_{i=0}^{2^k-1} x_i |i\rangle^{\otimes k} \right) |0\rangle^{\otimes k} \xrightarrow{UGD\ -\ Phase\ 2.} \sum_{i=0}^{2^k-1} x_i |i\rangle^{\otimes k} |i\rangle^{\otimes k}.$$

In phases three and four, unit operations of k qubits are performed in the first and second half of their respective qubits. An U operator is applied in terms of leftmost k qubits, and an V^T operator is applied in terms of rightmost k qubits. The application of these operators retrieve the original matrix that had been decomposed by the decomposition of singular values (SVD).

3.1.5 Flip-Flop QRAM Algorithm - FF-QRAM Algorithm

Proposed in (PARK; PETRUCCIONE; RHEE, 2019), the FF-QRAM is a probabilistic quantum state preparation algorithm capable of receiving as input a classical data set given by

$$\mathcal{D} = \left\{ (x_k, p_k) | x_k \in \mathbb{C}, \sum_k |x_k|^2 = 1, p_k \in \{0,1\}^n \right\}, \tag{3.4}$$

where $0 \leq k < M$ and $0 \leq j < n$, providing as output a computational base state superposition with continuous probability amplitudes to be used as state specific initial input required by a quantum algorithm.

Given a structure $\mathcal{D}$ with M patterns, FF-QRAM updates this M classical data into a superposition of quantum state by executing a quantum circuit consisting of $\mathcal{O}(n)$ qubits. Thus, the FF-QRAM algorithm needs at least $6n - 4$ CNOT gates at computational cost $\mathcal{O}(CnM)$, since FF-QRAM is probabilistic and required pos-selection, where C denotes the number of post-selections required, n is the number of qubits and M is the number of patterns (PARK; PETRUCCIONE; RHEE, 2019).

3.2 CLASS OF ALGORITHMS

A characteristic of the previously mentioned algorithms involves the preparation of the quantum state exactly as desired; that is, they are exact state preparation algorithms. FF-QRAM is a probabilistic algorithm that prepares an exact quantum state quantum with continuous amplitudes from a classical input data set $\mathcal{D}$. In quantum states preparation, PQM is a particular case of FF-QRAM since it prepares a quantum state with uniform amplitudes from a set of binary input data. Thus, we can establish that both the FF-QRAM and PQM algorithms belong to the same class of algorithms whose cost based on the number of CNOT gates depends on the number of n qubits and the number of M input patterns.

The UCR, SQL and UGD algorithms depend exclusively on the number of qubits n to prepare a quantum state of n qubits. We can establish that the UCR, SQL and UGD algorithms belong to the same class of algorithms whose cost based on the number of CNOT gates depends exclusively on the n number of qubits.

This work proposes two state preparation algorithms called CV-QRAM and CVO-QRAM. The CV-QRAM algorithm prepares an exact quantum state from a classical input data set $\mathcal{D}$ whose cost based on the number of CNOT gates depends on the size of input data set and the number of qubits. The CVO-QRAM algorithm prepares an exact quantum state from a classical input data set $\mathcal{D}$ whose CNOT cost depends on the size of the data set, the number of qubits, and the number of 1s in the binary string of each input pattern. Therefore, both the CV-QRAM and CVO-QRAM algorithms are exact quantum state preparation algorithms belonging to the same FF-QRAM and PQM algorithms class.

3.3 PREPARATION OF APPROXIMATE QUANTUM STATES

In (GROVER, 1997), Grover proposed a probabilistic algorithm of quantum exhaustive search, which is able to find, with high probability, an element in a list with $M = 2^n$ terms by using Walsh-Hadamard (WH) transformations (DEUTSCH; JOZSA, 1992), with a

cost of $\mathcal{O}(\sqrt{M})$ steps. This algorithm surpasses the classical search algorithms, which, in the worst scenario, cannot solve this problem at a cost below $\mathcal{O}(M)$ steps.

In (GROVER, 1998), Grover shows that by replacing the W-H transformations with suitable unit operators, very similar results are obtained. The contribution of this work is that upon a unit operation U and initializing the system in a base state, for example $|0\rangle^{\otimes n}$, it is possible to obtain a target quantum state $|target\rangle$ via $U|0\rangle^{\otimes n} = |target\rangle$.

In (GROVER, 2000), Grover shows how the quantum search algorithm can be generalized to solve the case of a multi-solution and consequently used to prepare an arbitrary quantum state, loading amplitudes in a quantum state and using a oracles as subroutines. The quantum state preparation based on the Grover's work can be described briefly as follows: Suppose we want to prepare a quantum state target

$$|\psi\rangle = \sum_{k=0}^{M-1} x_k |p_k\rangle_n,$$

where $x_k \in \mathbb{C}$, with $0 \leq x_k < 1$ for all k; $p_k \in \{0.1\}^n$, with $\sum_{i=0}^{k-1} |\alpha_i|^2 = 1$ using n qubits. The algorithm uses a quantum register called out, with n qubits, initialized in $|\psi_0\rangle = |0\rangle^{\otimes n}$, and prepares a quantum state containing the $M = 2^n$ states of the computational base. Then, a quantum oracle (black-box) called $\mathfrak{amp}$ is applied, whose output creates a new quantum register called data, with l qubits, a quantum state approximated to the desired state, given by

$$|\psi\rangle_{\mathfrak{amp}} = \frac{1}{\sqrt{M}} \sum_{k=0}^{M-1} |p_k\rangle_{out} \left|x_k^{(l)}\right\rangle_{data}, \tag{3.5}$$

where $x_k^{(l)}$ is a binary approximation of x_k amplitude, with $l \in \mathbb{N}$ bits of precision. Then, the subroutine defined is denoted by

$$\mathfrak{rot}\left|x_k^{(l)}\right\rangle|0\rangle = \left|x_k^{(l)}\right\rangle\Big(\sin\theta_k|0\rangle_{flag} + \cos\theta_k|1\rangle_{flag}\Big),$$

where the second record of a single qubit is called flag, $\theta_k \simeq \arcsin\left(\frac{x_k^{(l)}}{2^n}\right)$. This procedure performed a rotation in the single qubit called flag, controlled by arcsin. After applying this operator, we have the following quantum state

$$|\psi_{\mathfrak{rot}}\rangle = \frac{1}{\sqrt{M}} \sum_{k=0}^{M-1} |p_k\rangle_{out} \left|x_k^{(l)}\right\rangle_{data} \otimes \Big(\sin\theta_k|0\rangle_{flag} + \cos\theta_k|1\rangle_{flag}\Big)$$

The cost of the Grover's quantum state preparation algorithm is $\mathcal{O}(\sqrt{M})$ steps, amplifying the amplitudes performed by operators $\mathfrak{amp}$ and $\mathfrak{rot}$. Then, the data register is restarted by reapplying the operator $\mathfrak{amp}$ and can be discarded.

A measurement is performed on the flag register $\sin\theta_k|0\rangle_{flag} + \cos\theta_k|1\rangle_{flag}$, where $\sin\theta_k \simeq x_k$. If the result is $|1\rangle$, the algorithm has failed. On the other hand, if the result is $|0\rangle$, a quantum state close to the desired $|\psi\rangle$ has been prepared.

Thus, Grover established a new technique for quantum state preparation, called the Black-box quantum state preparation, involving the preparation of a quantum state approximate to the desired state. He was also the inspiration for other preparation works, such as (SOKLAKOV; SCHACK, 2005), (SANDERS et al., 2019), (BAUSCH, 2020).

As a Grover approach, it required the quantum computer to calculate the angles of rotation for each input value using arcsine. The Algorithm proposed in (SANDERS et al., 2019) promoted a change in Grover's approach, starting from the state $\psi_{\mathfrak{amp}}$ given by Eq. (3.5), eliminating the operator $\mathfrak{rot}$, whose technical subroutine the quantum computer calculate, for each k, the angles θ_k, such that $\sin\theta_k \simeq x_k$.

Thus, the attribution of new operators eliminated the need for arithmetic, in which the amplitude transduction started to be performed by testing an inequality between an input value and all possible overlapping output values. In the end of the test results and after amplifying the amplitude, the input value requires to be transduced as an amplitude. With this change, in which arithmetic is replaced with a inequality test, the authors expected to obtain significantly less complexity of quantum simulation algorithms.

(BAUSCH, 2020) also proposes a change in the algorithm established by Grover, after creating the quantum state given by eq.(3.5), the authors estimate how far this approximate state is from the desired real quantum state, using the quantum gradient state, which can store the approximate amplitude values with high precision. This paper has achieved better results than in (SANDERS et al., 2019), exponentially reducing the number of required qubits and maintaining the distance accuracy between the approximate and target states, in addition to decreasing the number of necessary amplification steps.

The algorithms particularly mentioned in this section are outside the scope of our work, since our algorithms prepare the state exactly the same as desired, that is, they are algorithms for preparing the exact states. Whereas the black-box state preparation algorithms prepare quantum states approximately equal to the desired state, and are called approximate state preparation algorithms.

4 RESULTS OF THE PROPOSED ALGORITHMS

This chapter introduces the main contributions for quantum states preparation achieved in the course of this dissertation. As mentioned earlier, both the PQM and FF-QRAM were the main algorithms on which our research was based.

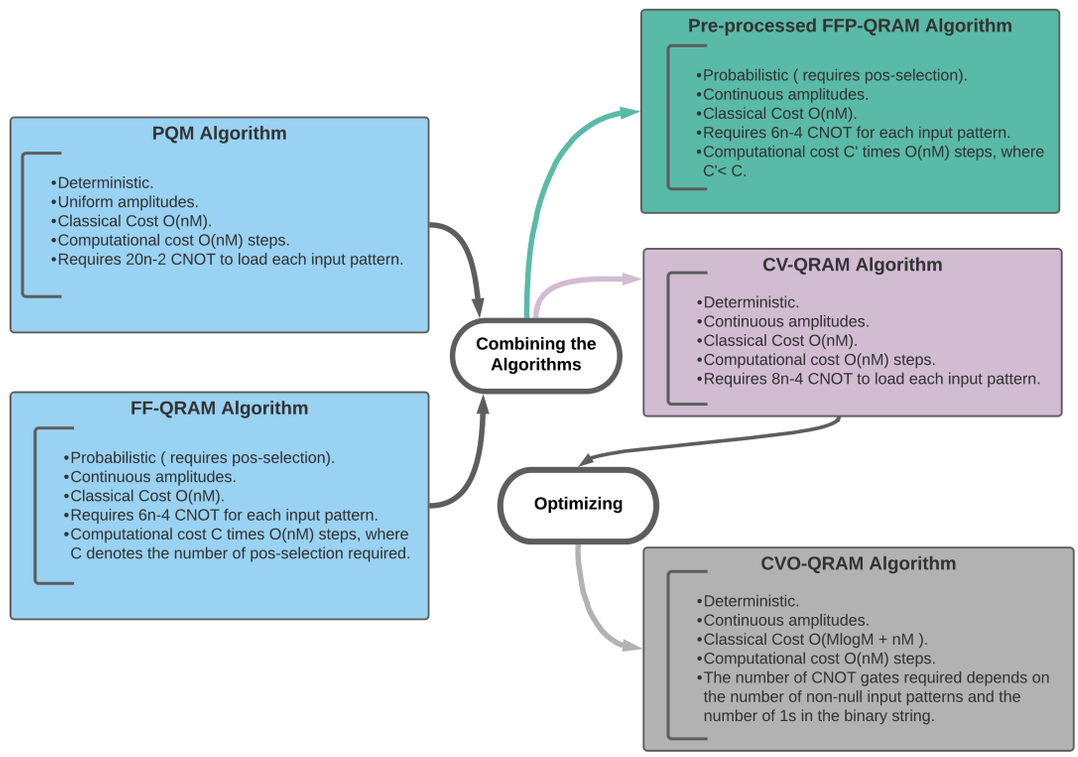
Combining the PQM and FF-QRAM algorithms, we built the Pre-processed FFP-QRAM, which can improve the post-selection of FF-QRAM by performing a strategic pre-processing on the input dataset, followed by the preparation of the initial quantum state by PQM.

From the combination of the PQM and FF-QRAM algorithms, we built our second quantum state preparation algorithm called CV-QRAM. The main goal of building the CV-QRAM to eliminate the post-selection existing in the FF-QRAM.

Aiming to reduce the number of CNOT gates needed to prepare the quantum state, we propose the CVO-QRAM, an optimization of the CV-QRAM. CVO-QRAM obtained better results than CV-QRAM in all scenarios in the preparation of states.

The diagram in Figure (5) summarizes the construction path of the Pre-processed FFP-QRAM, CV-QRAM, and CVO-QRAM algorithms built in this dissertation research, which will be presented in two articles in sections 4.1 and 4.2.

Figure 5 – Diagram of the construction path of the algorithms obtained in this dissertation research.



Source: The author (2020)

4.1 CIRCUIT-BASED QUANTUM RANDOM ACCESS MEMORY FOR CLASSICAL DATA WITH CONTINUOUS AMPLITUDES

This section presents our first two quantum state preparation algorithms, which we designate as Pre-processed FFP-QRAM and CV-QRAM. Pre-processed FFP-QRAM is probabilistic and aimed at improving the existing post-selection in FF-QRAM. Initially, we performed a strategic pre-processing on the amplitudes of the input dataset, the initial state of the Pre-processed FFP-QRAM is prepared by the PQM. The Pre-processed FFP-QRAM algorithm is the FF-QRAM executed with strategic pre-processing, whose initial state is the state prepared by PQM. This allowed us to develop an algorithm that solves our first specific goal: to improve FF-QRAM post-selection.

Based on the previous works on PQM (TRUGENBERGER, 2001) and FF-QRAM (PARK; PETRUCCIONE; RHEE, 2019), the CV-QRAM (VERAS et al., 2020) was built to prepare a quantum state with continuous amplitudes, without requiring post-selection. By combining the PQM and FF-QRAM algorithms, we were able to build an algorithm that has computational cost $\mathcal{O}(Mn)$ steps, without demanding post-selection, carrying amplitudes to continue with linear classical cost $\mathcal{O}(nM)$. Thus, the CV-QRAM solves our second specific goal and is published in IEEE Transactions on Computers (VERAS et al., 2020). In the following text, CV-QRAM is described as A-PQM.

Circuit-based quantum random access memory for classical data with continuous amplitudes

Tiago M. L. de Veras, Ismael C. S. de Araujo, Daniel K. Park, Adenilton J. da Silva,

Abstract

Loading data in a quantum device is required in several quantum computing applications. Without an efficient loading procedure, the cost to initialize the algorithms can dominate the overall computational cost. A circuit-based quantum random access memory named FF-QRAM can load M n-bit patterns with computational cost $O(CMn)$ to load continuous data where C depends on the data distribution. In this work, we propose a strategy to load continuous data without post-selection with computational cost $O(Mn)$. The proposed method is based on the probabilistic quantum memory, a strategy to load binary data in quantum devices, and the FF-QRAM using standard quantum gates, and is suitable for noisy intermediate-scale quantum computers.

Index Terms

Quantum RAM, Quantum state initialization, Data loading in quantum devices

1 Introduction

Quantum computation [1], [2] has the potential to speed up the solution of certain computational problems. These speedups are due to the inherent properties of quantum mechanics, such as superposition, entanglement, and interference. The power of quantum computation over its classical counterpart has been theoretically demonstrated in several problems, such as simulating quantum systems [2], [3], unstructured data search [4], prime factorization [5], machine learning [6]. However, the development of full-fledged quantum hardware capable of operating efficiently on these problems remains unsolved.

A desideratum for practical and wide application of quantum algorithms is the efficient means to load and update classical data in a quantum computer [7]. In other words, a programmer needs to be able to encode the input data structured as

$$\mathcal{D} = \left\{ (x_k, p_k) | x_k \in \mathbb{C}, \sum_k |x_k|^2 = 1, p_k \in \{0,1\}^n \right\}, \tag{1}$$

where $0 \leq k < M$, into a quantum state efficiently. In addition, $p_k[j]$ denotes jth bit of a pattern p_k, with $0 \leq j < n$.

The classical data can be represented as a quantum state

$$|\psi\rangle = \sum_{k=0}^{M-1} x_k |p_k\rangle \tag{2}$$

using n qubits. When $n = \log_2(M)$, this is equivalent to amplitude encoding [8], which achieves exponential data compression. Hereinafter, we refer to the data representation in the form of Eq. (2) as generalized amplitude encoding (GAE). The resource overhead, such as the number of qubits and the circuit depth, for preparing the quantum data can dominate the overall computational cost due to the quantum measurement postulate — one often needs to repeat the same algorithm multiple times to gather measurement statistics while each measurement destroys the quantum state. Moreover, the state initialization must be carried out substantially faster than the decoherence time. Therefore, a fast algorithm for initializing the quantum data is of critical importance, and this is the problem that we address in this manuscript. The

- *T. M. L. de Veras is with Centro de Informática, Universidade Federal de Pernambuco and Departamento de Matemática, Universidade Federal Rural de Pernambuco, Recife, Pernambuco, Brazil.*
- *I.C.S. de Araujo is with Centro de Informática, Universidade Federal de Pernambuco and the Departamento de Computação, Universidade Federal Rural de Pernambuco, Recife, Pernambuco, Brazil.*
- *D. K. Park is with School of Electrical Engineering, Korea Advanced Institute of Science and Technology, Daejeon, Korea.*
- *A.J. da Silva is with Centro de Informática, Universidade Federal de Pernambuco, Recife, Pernambuco, Brazil.*
  E-mail: ajsilva@cin.ufpe.br

data loading procedure should be designed with quantum circuit elements since the circuit model provides systematic and efficient instructions to achieve universal quantum computation.

The initialization of quantum data can be achieved by utilizing a quantum random access memory (QRAM). The bucket brigade model for QRAM introduced in Ref [9] uses $O(M)$ quantum hardware components and $O(\log_2^2(M))$ time steps to return binary data stored in M memory cells in superposition. This procedure can be expressed as

$$\frac{1}{\sqrt{M}}\sum_{k=0}^{M-1}|k\rangle|0\rangle \xrightarrow{QRAM} \frac{1}{\sqrt{M}}\sum_{k=0}^{M-1}|k\rangle|p_k\rangle. \tag{3}$$

Unfortunately, translating this model to the quantum circuit model while retaining its advantage is difficult [10], [11]. Moreover, the full state initialization requires additional steps. First, controlled rotations are applied to prepare

$$\frac{1}{\sqrt{M}}\sum_{k=0}^{M-1}|k\rangle|p_k\rangle\left(\sqrt{1-|x_k|^2}|0\rangle + x_k|1\rangle\right). \tag{4}$$

Then by uncomputing $|p_k\rangle$ and post-selecting the last register onto $|1\rangle$ the amplitude encoding is completed [12]. Similarly, the GAE can be achieved by uncomputing $|k\rangle$ instead.

Ref. [13] introduced a QRAM architecture that is constructed with the standard quantum circuit elements to provide flexibility and compatibility with existing quantum computing techniques. This model registers the classical data structure $\mathcal{D}$ with M patterns into quantum format by repeating the execution of a quantum circuit consisting of $O(n)$ qubits and $O(Mn)$ flip-register-flop operations C times, and hence called flip-flop QRAM (FF-QRAM). The repetition is necessary to post-select the correct outcome for encoding continuous data (similar to explanations around Eq. (4). When the FF-QRAM was first introduced, the extra cost for post-selection was neglected. However, C can be a function of M and n, depending on the given classical data. Therefore, designing a QRAM for general classical data without the post-selection procedure, or with a minimal number of repetitions, remains a critical open problem. On the other hand, when storing a set of binary patterns into a quantum state with equal probability amplitudes, the post-selection is not necessary. The probabilistic quantum memory (PQM) [14] is a viable method to achieve this. Both FF-QRAM and PQM are not fully satisfactory as the former requires expensive post-selection, and the latter can only encode binary data.

The main contribution of this work is twofold. First, we show that the number of repetitions for post-selection in the FF-QRAM algorithm can be reduced by using the PQM as a preliminary step of the FF-QRAM and preprocessing the classical data. Second, we present a deterministic quantum data preparation algorithm based on FF-QRAM and PQM that achieves the generalized amplitude encoding without post-selection.

The rest of this manuscript is structured as follows. Section 2 gives a brief description of the main quantum gates used in this work. Section 3 provides a brief review on FF-QRAM [13] and PQM [14], the previous works by which the current work is inspired. Section 4 shows that in the worst-case FF-QRAM has an exponential computational cost due to post-selection. Section 5 presents the main results: methods to improve the post-selection probability and to remove the need for post-selection. Section 6 presents proof-of-principle experiments to demonstrate the improvement achieved by our methods, and Section 7 draws the conclusion.

2 Quantum Operators

This section is dedicated to define unitary operators that will appear in this manuscript besides the well-known gates, such as I, X and $R_y(\theta)$ [15].

For controlled unitary gates, the control and the target will be indicated by a subscript, separated by a comma, in which the control qubit appears first. Given a quantum state $|\psi\rangle$ containing bits x_0 and x_1, we use

$$CX_{x_0,x_1}|\psi\rangle \tag{5}$$

to indicate a controlled-X operation, which transforms the target $|x_1\rangle$ to $|x_1 \oplus x_0\rangle$.

Equation (5) can be generalized to express a gate that is controlled by multiple qubits. Given a quantum state $|\psi\rangle$, containing $n+1$ bits $x_0, x_1, \ldots, x_{n-1}, x_n$, we use

$$C^nX_{(x_0x_1\ldots x_{n-1}),x_n}|\psi\rangle \tag{6}$$

to denote an n-qubit controlled-X operation, which transforms the target $|x_n\rangle$ to $|x_n \oplus x_0 \cdot x_1 \cdot \ldots \cdot x_{n-1}\rangle$.

This work also uses single-qubit rotations controlled by n qubits, denoted by $C^nR_a(\theta)$, where a is one of the axes x, y or z. Note that a multi-qubit controlled unitary gate can be decomposed to $n-1$ Toffoli gates and a single-qubit controlled unitary gate using $n-1$ ancillae [15].

Similarly, a bit-flip operator controlled by a classical bit can be defined as

$$cX_{x_0,x_1} \tag{7}$$

which applies X to $|x_1\rangle$ if $x_0 = 1$. The FF-QRAM algorithm uses a bit-flip operator that flips the target qubit if the classical control bit is 0. In this case, we denote the operator with an overline as $\bar{c}X_{x_0,x_1}$.

Another important operator in this work is a gate controlled by classical and quantum states as follows. Given a classical bit p, this operation works as

- If $p = 0$, do $X|m\rangle$.
- If $p = 1$, do $CX_{u_2,m}|u_2 m\rangle$,

and its quantum circuit representation is depicted in Figure 1.

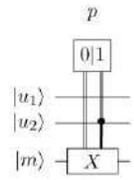


Fig. 1: A quantum circuit representation of the classical-quantum controlled gate that applies X to $|m\rangle$ if $p = 0$, and applies $CX_{u_2,m}$ if $p = 1$.

The PQM algorithm uses a controlled unitary operator denoted by $CS^r_{x_0,x_1}$, which applies

$$S^r = \begin{bmatrix} \sqrt{\frac{r-1}{r}} & \frac{1}{\sqrt{r}} \\ \frac{-1}{\sqrt{r}} & \sqrt{\frac{r-1}{r}} \end{bmatrix}, \tag{8}$$

where $r \in \mathbb{N}$, to the target x_1 if $x_0 = 1$.

Now, given a single-qubit unitary matrix

$$U_3 = \begin{bmatrix} \cos\frac{\theta}{2} & -e^{i\lambda}\sin\frac{\theta}{2} \\ e^{i\phi}\sin\frac{\theta}{2} & e^{i(\lambda+\phi)}\cos\frac{\theta}{2} \end{bmatrix}, \tag{9}$$

it is possible to obtain θ, λ and ϕ, such that U_3 can be rewritten as a matrix with two parameters, x_k and γ_k, as follows

$$U_3^{(x_k,\gamma_k)} = \begin{bmatrix} \sqrt{\frac{\gamma_k - |x_k|^2}{\gamma_k}} & \frac{x_k}{\sqrt{\gamma_k}} \\ \frac{-x_k^*}{\sqrt{\gamma_k}} & \sqrt{\frac{\gamma_k - |x_k|^2}{\gamma_k}} \end{bmatrix}. \tag{10}$$

To obtain (10), it is necessary that:

(i) $\cos\frac{\theta}{2} = \sqrt{\frac{\gamma_k - |x_k|^2}{\gamma_k}} \iff \theta = 2\arccos\sqrt{\frac{\gamma_k - |x_k|^2}{\gamma_k}}$.

(ii) $-e^{i\lambda}\sin\frac{\theta}{2} = -(\cos\lambda + i\sin\lambda)\sin\frac{\theta}{2} = \frac{x_k}{\sqrt{\gamma_k}}$.
This admits one of the following solutions:
   a) $\lambda = \arccos\left(-\frac{\sqrt{a}}{\sqrt{\gamma_k}\sin\frac{\theta}{2}}\right)$
   b) $\lambda = \arcsin\left(-\frac{\sqrt{b}}{\sqrt{\gamma_k}\sin\frac{\theta}{2}}\right)$.

(iii) Therefore $\phi = -\lambda$ is the solution to $e^{i(\lambda+\phi)}\cos\frac{\theta}{2}$.

The unitary operator $U_3^{(x_k,\gamma_k)}$ shown in Eq. (10) is the key ingredient in this work, inspired by S^r, where $\gamma_k = \gamma_{k-1} - |x_{k-1}|^2$ is a complex iteration variable with $1 \leqslant k \leqslant M-1$, $\gamma_0 = 1$, and $M \in \mathbb{N}$. Furthermore, $x_k = \sqrt{a} + i\sqrt{b}$ is a complex number with its complex conjugate denoted by x_k^*. While CS^r can load binary data only, the controlled operation of $U_3^{(x_k,\gamma_k)}$ enables the loading of complex data.

3 Related works

In this section, we briefly review two storage algorithms that motivated this work, namely FF-QRAM and PQM. We point readers to Refs. [13] and [14], [16] for more details on FF-QRAM and PQM, respectively.

3.1 Flip-Flop Quantum RAM

The objective of the FF-QRAM algorithm [13] is to load classical data $\mathcal{D}$ with M patterns shown in Eq. (1) to a quantum state shown in Eq. (2) using a quantum circuit. For this, the quantum circuit must receive as the initial state of the algorithm

$$|\psi_0\rangle = \sum_{t=0}^{2^n-1} \alpha_t |t\rangle_B |0\rangle_R, \tag{11}$$

where B and R indicate n address qubits and a register with one qubit, respectively, $|t\rangle$ is a computational basis of n qubits, $\alpha_t \in \mathbb{C}$, and $M \leq 2^n$.

For the input (x_k, p_k), we can associate the following initial state:

$$|\psi_0\rangle_k = \alpha_k |p_k\rangle_B |0\rangle_R + \sum_{t\neq k} \alpha_t |t\rangle_B |0\rangle_R. \tag{12}$$

To obtain the state $|\psi_1\rangle_k$, the controlled bit-flip operator $\bar{c}X$, controlled by a classical input string p_k, is applied to the address qubits (indicates by B) of $|\psi_0\rangle_k$. The $\bar{c}X$ operation flips the target qubit if the control bit is 0. This is the *flip* operation that results in

$$|\psi_1\rangle_k = \alpha_k |1\rangle_B^{\otimes n} |0\rangle_R + \sum_{|\overline{t\oplus p_k}\rangle \neq |1\rangle^{\otimes n}} \alpha_t \big|\overline{t\oplus p_k}\big\rangle_B |0\rangle_R, \tag{13}$$

with $|\bar{i}\rangle$ being the ith negated binary string. The terms with an overline means that each bit is flipped if the control bit is zero. In the next step, a multi-qubit controlled operator that applies $R_y(\theta_k) = \cos(\theta_k/2)I - i\sin(\theta_k/2)Y$ on the register qubit $|0\rangle_R$ if the n-qubit address state is $|1\rangle^{\otimes n}$ yields the state

$$|\psi_2\rangle_k = \alpha_k |1\rangle^{\otimes n} |\theta_k\rangle_R + \sum_{|\overline{t\oplus p_k}\rangle \neq |1\rangle^{\otimes n}} \alpha_t \big|\overline{t\oplus p_k}\big\rangle |0\rangle_R, \tag{14}$$

where $|\theta_k\rangle = \cos(\theta_k/2)|0\rangle + \sin(\theta_k/2)|1\rangle$. The angle θ_k is chosen based on x_k. Note that Ref. [13] does not discuss explicitly on how to find the angle. Thus, in this section we show the condition necessary to obtain this angle. Later, in an example, we will show how the angle can be found. Next, $\bar{c}X$ is applied again on the address qubits with the controlling bits being p_k. This is the *flop* operation that results in

$$|\psi_3\rangle_k = \alpha_k |p_k\rangle_B |\theta_k\rangle_R + \sum_{t\neq k} \alpha_t |t\rangle_B |0\rangle_R. \tag{15}$$

We are ready to go through the algorithm again and to load the next input (x_{k+1}, p_{k+1}). Proceeding in a similar way to the previous step and applying the flip-flop operation with p_{k+1} and the controlled-rotation with the angle $\theta_{(k+1)}$, we obtain

$$\begin{aligned} |\psi_4\rangle_{k,k+1} =& \alpha_k |p_k\rangle_B |\theta_k\rangle_R + \alpha_{k+1} |p_{k+1}\rangle_B \big|\theta_{(k+1)}\big\rangle_R \\ &+ \sum_{t\neq k,k+1} \alpha_t |t\rangle_B |0\rangle_R. \end{aligned} \tag{16}$$

Performing this procedure M times, all M inputs are registered in the quantum state as follows.

$$\sum_{k=0}^{M-1} \alpha_k |p_k\rangle [\cos(\theta_k/2)|0\rangle + \sin(\theta_k/2)|1\rangle] + \sum_{t\neq k} \alpha_t |t\rangle |0\rangle_R. \tag{17}$$

To complete the generalized amplitude encoding, firstly, we need to get rid of the terms where the register qubit is in $|0\rangle_R$. So we must obtain a convenient θ_k, such that the probability of amplitude for state $|1\rangle$ is high. Then, we will evaluate

$$P(|1\rangle) = \sum_{k=0}^{M-1} \Big|\alpha_k \sin(\theta_k/2)\Big|^2. \tag{18}$$

At this point, it is clear that the angle θ_k must satisfy the condition

$$\frac{\alpha_k \sin(\theta_k/2)}{\sqrt{P(|1\rangle)}} = x_k. \tag{19}$$

The FF-QRAM algorithm takes at least $O(nM)$ steps for each run of the quantum circuit, and there is an additional cost for post-selection [13]. If $P(|1\rangle) \approx 1$, then the desired state shown in Eq. (2) is obtained with a high probability. Otherwise, the additional cost for post-selection can be non-negligible. We will show in Section 4 that this additional cost can compromise the efficiency of the algorithm.

The FF-QRAM algorithm is summarized in Algorithm 1, and a quantum circuit for storing a particular data (x_k, p_k) is depicted in Figure 2.

Algorithm 1: FF-QRAM

input : data = $\{(x_k, p_k)\}_{k=0}^{M-1}$, $|\psi_0\rangle = \sum_t \alpha_t |t\rangle|0\rangle$
output: $|\psi\rangle = \sum_{k=0}^{M-1} x_k |p_k\rangle$, s

1 `load` ($data$, $|\psi\rangle$)**:**
2 **foreach** $(x_k, p_k) \in data$ **do**
3 $\prod_{j=0}^{n-1} \bar{c}X_{p_k[j],m[j]}|\psi\rangle$
4 $C^n Ry_{R,B}(\theta_k)|\psi\rangle_R|0\rangle_B$
5 $\prod_{j=0}^{n-1} \bar{c}X_{p_k[j],m[j]}|\psi\rangle$
6 $s = measure_B|\psi\rangle|B\rangle$
7 **return** $|\psi\rangle$, s

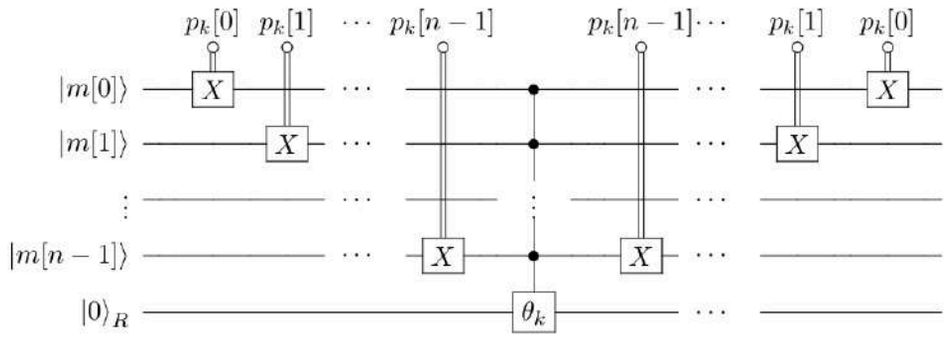


Fig. 2: One iteration of the FF-QRAM data storage algorithm.

3.2 Loading binary patterns

The PQM [14], [16] initialization algorithm is based on [17], [18], and is capable of storing a classical binary dataset $data = \cup_{k=0}^{M-1}\{p_k\}$, where p_k is an n-bit pattern (i.e. $p_k \in \{0,1\}^n$), in a quantum state as

$$|M\rangle = \frac{1}{\sqrt{M}} \sum_{k=0}^{M-1} |p_k\rangle, \tag{20}$$

with computational cost is $O(nM)$.

The storage algorithm uses three quantum registers. The first register is $|p\rangle$, with n qubits to which the target pattern p_k is loaded in each iteration of the algorithm. The second is an auxiliary register $|u\rangle = |u_1 u_2\rangle$ with two qubits initialized in state $|01\rangle$. The final register of n qubits holds the memory, is denoted by $|m\rangle$, and $m[j]$ denotes jth qubit of $|m\rangle$. The initial state of the total system is

$$|\psi_0\rangle = |0\ldots0;01;0,\ldots0\rangle. \tag{21}$$

The underlying idea of the algorithm is to split this state into two parts in each iteration. One corresponds to the subspaces of the total quantum state with patterns stored in the memory register, and another to process a new pattern. These two parts are distinguished by the second qubit of the auxiliary register, u_2, which is 0 for the subspaces that already store patterns and 1 for the subspace to which a new pattern is being loaded (are processing terms). After storing a pattern, the first qubit of the auxiliary register, $|u_1\rangle$, is used to apply a controlled operator on $|u_2\rangle$ to split the state for storing the next pattern.

Algorithm 2 presents the PQM storage algorithm named `load_binary`. In the PQM storage algorithm, we use $|\psi_{k_i}\rangle$ to denote the quantum state in step i of storing the pattern p_k.

In the step 5, the transformation

$$|\psi_{k_1}\rangle = \prod_{j=0}^{n-1} C^2 X_{(p_k[j]u_2),m_j}|\psi_{k_0}\rangle, \tag{22}$$

copies the binary pattern p_k from $|p\rangle$ register to the memory register $|m\rangle$. Note that these operations are applied to the jth memory qubit state $m[j]$ only if $p_k[j]$ and u_2 are in $|1\rangle$. In the step 6, the transformation

$$|\psi_{k_2}\rangle = \prod_{j=0}^{n-1} X_{m_j} C X_{p_k[j],m[j]}|\psi_{k_1}\rangle, \tag{23}$$

Algorithm 2: Probabilistic quantum memory storage algorithm

input : data = $\{p_k\}_{k=0}^{M-1}$

output: $|\psi\rangle = \frac{1}{\sqrt{M}} \sum_{k=0}^{M-1} |p_k\rangle$

1 load_binary ($data$, $|\psi\rangle$):
2 Initial state $|\psi_0\rangle = |0, 0, \cdots, 0; 01; 0, 0, \cdots, 0\rangle$
3 **foreach** $p_k \in data$ **do**
4 $|\psi_{k_0}\rangle =$ Load p_k into quantum register $|p\rangle$
5 $|\psi_{k_1}\rangle = \prod_{j=0}^{n-1} C^2X_{(p_k[j]u_2),m_j}|\psi_{k_0}\rangle$
6 $|\psi_{k_2}\rangle = \prod_{j=0}^{n-1} X_{m_j} CX_{p_k[j],m_j}|\psi_{k_1}\rangle$
7 $|\psi_{k_3}\rangle = C^nX_{m,u_1}|\psi_{k_2}\rangle$
8 $|\psi_{k_4}\rangle = CS_{u_1,u_2}^{M-k}|\psi_{k_3}\rangle$
9 $|\psi_{k_5}\rangle = C^nX_{m,u_1}|\psi_{k_4}\rangle$
10 $|\psi_{k_6}\rangle = \prod_{j=0}^{n-1} CX_{p_k[j],m_j}X_{m_j}|\psi_{k_5}\rangle$
11 $|\psi_{k_7}\rangle = \prod_{j=0}^{n-1} C^2X_{(p_k[j]u_2),m_j}|\psi_{k_6}\rangle$
12 Unload p_k from quantum register $|p\rangle$
13 **return** $|\psi\rangle$

yields $m[j] = 1$, if $p_k[j] = m[j]$, otherwise $m[j] = 0$. In the step 7,

$$|\psi_{k_3}\rangle = C^nX_{m,u_1}|\psi_{k_2}\rangle. \tag{24}$$

Here, we have an operation controlled by n bits of the memory, that is, if $m[j] = 1$ for all values of j, then the bit-flip gate X is applied to the bit u_1. This makes the qubit u_1 to assume the value 1 for the term in processing. Step 8,

$$|\psi_{k_4}\rangle = CS_{u_1,u_2}^{M-k}|\psi_{k_3}\rangle, \tag{25}$$

is the central operation of the storing algorithm, which separates the new pattern to be stored, with the correct normalization factor. The steps 9 and 10 are the inverse operators of steps 7 and 6, respectively. These will reset the values of u_1 and m to the initial values. This results in the following state:

$$|\psi_{k_6}\rangle = \frac{1}{\sqrt{M}} \sum_{s=1}^{k} |p_k; 00; p_s\rangle + \frac{\sqrt{M-k}}{\sqrt{M}} |p_k; 01; p_k\rangle \tag{26}$$

The step 11,

$$|\psi_{k_7}\rangle = \prod_{j=0}^{n-1} C^2X_{(p_k[j]u_2),m_j}|\psi_{k_6}\rangle, \tag{27}$$

resets the memory record of the term being processed ($u_2 = 1$) to to $|m\rangle = |0\rangle$.

At this point, we are ready to store the next pattern p_{k+1}. To do this, define $|\psi_{k+1}\rangle_0$ from $|\psi_{k_7}\rangle$ with the pattern p_{k+1} loaded in the register $|p\rangle$, and perform a new iteration of the algorithm. The process is iterated until $|u_2\rangle = |0\rangle$ in all terms of the quantum state, indicating that all patterns p_k are stored in the memory. Note that although the loading procedure is deterministic, memory read-out procedures are probabilistic due to the quantum measurement postulates. Thus this is the initialization algorithm of a *probabilistic* quantum memory.

Given that the patterns to be stored are of n bits, steps 4 and 12 require the same number of steps, $O(n)$. Then for M patterns to be stored, the entire algorithm requires $O(nM)$ steps to store all patterns. The circuit corresponding to one iteration of the PQM algorithm is shown in Figure 3.

In what follows, we demonstrate how to use the algorithm above to store an example set of patterns

$$data = \{p_0 = 00, p_1 = 01\}.$$

First iteration stores the pattern $p_0 = 00$. Loading the pattern in the initial state by step 4 results in

$$|\psi_{0_0}\rangle = |00; 01; 00\rangle.$$

The step 5 gives $|\psi_{0_0}\rangle = |\psi_{0_1}\rangle$, and by step 6,

$$|\psi_{0_2}\rangle = |00; 01; 11\rangle.$$

By step 7,

$$|\psi_{0_3}\rangle = |00; 11; 11\rangle.$$

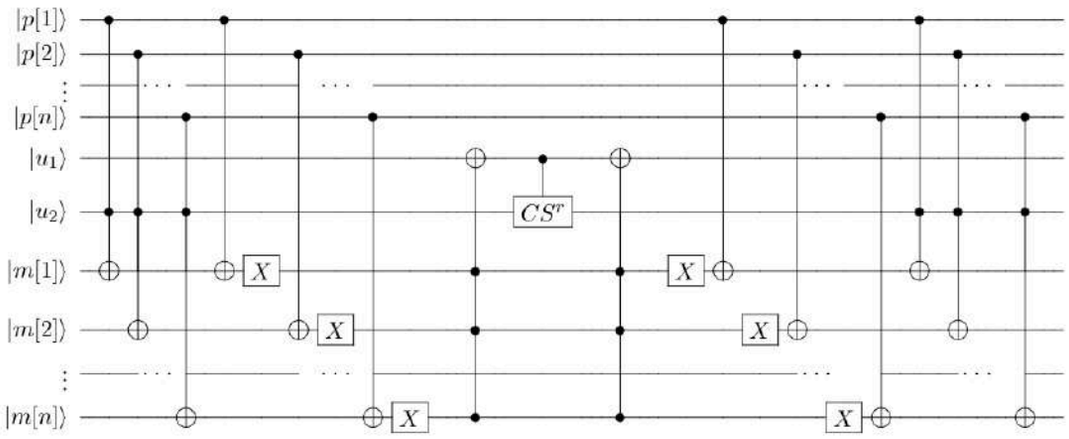


Fig. 3: One iteration of the PQM data storage algorithm.

Now, for $k = 0$, $r = 2$. Hence in step 8,

$$S^2 = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}.$$

Following step 8, which calculates $CS^2|11\rangle$, the state becomes

$$|\psi_{0_4}\rangle = \frac{1}{\sqrt{2}}|00;10;11\rangle + \frac{1}{\sqrt{2}}|00;11;11\rangle. \tag{28}$$

Applying steps 9 and 10 resets u_1 and the memory register, obtaining, respectively:

$$|\psi_{0_5}\rangle = \frac{1}{\sqrt{2}}|00;00;11\rangle + \frac{1}{\sqrt{2}}|00;01;11\rangle, \tag{29}$$

and

$$|\psi_{0_6}\rangle = \frac{1}{\sqrt{2}}|00;00;00\rangle + \frac{1}{\sqrt{2}}|00;01;00\rangle. \tag{30}$$

By step 11, we have $|\psi_{0_6}\rangle = |\psi_{0_7}\rangle$.

At this point, the quantum state has two terms. The first term has the auxiliary qubit $u_2 = 0$, which indicates that the pattern $p_0 = 00$ is stored in memory m. On the other hand, in the second term, $u_2 = 1$. This means that the term is in processing and can receive the next pattern to be stored at the memory. Now, we will do a new iteration in the algorithm to store the pattern $p_1 = 01$.

By step 4, as $|\psi_{1_0}\rangle$ is equal to $|\psi_{0_7}\rangle$ with the pattern p_1 loaded in the register, the quantum state becomes

$$|\psi_{1_0}\rangle = \frac{1}{\sqrt{2}}|01;00;00\rangle + \frac{1}{\sqrt{2}}|01;01;00\rangle.$$

Then, by steps 5, 6 and 7, we have respectively:

$$|\psi_{1_1}\rangle = \frac{1}{\sqrt{2}}|01;00;00\rangle + \frac{1}{\sqrt{2}}|01;01;01\rangle,$$

$$|\psi_{1_2}\rangle = \frac{1}{\sqrt{2}}|01;00;10\rangle + \frac{1}{\sqrt{2}}|01;01;11\rangle,$$

and

$$|\psi_{1_3}\rangle = \frac{1}{\sqrt{2}}|01;00;10\rangle + \frac{1}{\sqrt{2}}|01;11;11\rangle.$$

Now, for $k = 1$, $r = 1$. Hence in step 8,

$$S^1 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}.$$

Following step 8, which results in $CS^1|00\rangle = |00\rangle$ and $CS^1|11\rangle = |10\rangle$, the state becomes

$$|\psi_{1_4}\rangle = \frac{1}{\sqrt{2}}|01;00;10\rangle + \frac{1}{\sqrt{2}}|01;10;11\rangle.$$

Applying steps 9 and 10 resets u_1 and the memory register, respectively:

$$|\psi_{1_5}\rangle = \frac{1}{\sqrt{2}}|01;00;10\rangle + \frac{1}{\sqrt{2}}|01;00;11\rangle,$$

and

$$|\psi_{1_6}\rangle = \frac{1}{\sqrt{2}}|01;00;00\rangle + \frac{1}{\sqrt{2}}|01;00;01\rangle.$$

Finally, as $|\psi_{1_6}\rangle = |\psi_{1_7}\rangle$ and $u_2 = 0$ in all terms of $|\psi_{1_7}\rangle$, there are no more processing terms and we conclude that the binary patterns are stored in the memory m, producing the state

$$|M\rangle = \frac{1}{\sqrt{2}}|01;00;00\rangle + \frac{1}{\sqrt{2}}|01;00;01\rangle. \tag{31}$$

This is the quantum state desired in Eq. (20) for the example data set.

4 LIMITATION IN THE FF-QRAM ALGORITHM

In this section, we show that the FF-QRAM algorithm becomes inefficient if the initial state in Eq. (11) is used to load $v \ll M$ patterns as in the original design. More specifically, the post-selection can make the algorithm a non-viable choice for efficiently storing real amplitudes in a quantum state. For this, consider the following list of continuous data:

$$\left\{\left(\sqrt{0.3}, |000\rangle\right), \left(\sqrt{0.7}, |001\rangle\right)\right\},$$

meaning that $x_0 = \sqrt{0.3}$, $x_1 = \sqrt{0.7}$, $n = 3$, and $M = 2$. The initial state is simply $|+\rangle_B^{\otimes 3}|0\rangle_R$.

After step 3 of the first iteration in Algorithm 1,

$$\begin{aligned}|\psi_1\rangle_0 = \frac{1}{2\sqrt{2}}\Big(&|111\rangle + |110\rangle + |101\rangle + |100\rangle \\ &+|011\rangle + |010\rangle + |001\rangle + |000\rangle\Big)|0\rangle_R.\end{aligned} \tag{32}$$

In the next step, a controlled-rotation, $C^3R_y(\theta_0)$, is applied to rotate the register qubit that is entangled with $|111\rangle_B$:

$$\begin{aligned}|\psi_2\rangle_{x_0} =&\frac{1}{2\sqrt{2}}|111\rangle\Big[\cos\left(\frac{\theta_0}{2}\right)|0\rangle + \sin\left(\frac{\theta_0}{2}\right)|1\rangle\Big] \\ &+\frac{1}{2\sqrt{2}}\sum_{|t\rangle\neq|111\rangle}|t\rangle|0\rangle_R\end{aligned} \tag{33}$$

With $\theta_0 = 2\arcsin\sqrt{0.3} = 1.159$, Eq. (33) becomes

$$\begin{aligned}|\psi_2\rangle_{x_0} = &\frac{1}{2\sqrt{2}}|111\rangle\Big(\sqrt{0.7}|0\rangle + \sqrt{0.3}|1\rangle\Big) \\ &+\frac{1}{2\sqrt{2}}\Big(|110\rangle + |101\rangle + |100\rangle + |011\rangle \\ &+\ |010\rangle + |001\rangle + |000\rangle\Big)|0\rangle_R.\end{aligned} \tag{34}$$

Step 5 in Algorithm 1 completes the first iteration to produce

$$\begin{aligned}|\psi_3\rangle_{x_0} = &\frac{1}{2\sqrt{2}}|000\rangle\Big(\sqrt{0.7}|0\rangle + \sqrt{0.3}|1\rangle\Big) \\ &+\frac{1}{2\sqrt{2}}\Big(|001\rangle + |010\rangle + |011\rangle + |100\rangle \\ &+\ |101\rangle + |110\rangle + |111\rangle\Big)|0\rangle_R.\end{aligned} \tag{35}$$

Similar procedure is followed to load $\left(\sqrt{0.7}, |001\rangle\right)$. After repeating up to step 3 in Algorithm 1, the state becomes

$$\begin{aligned}|\psi_4\rangle_{x_0,x_1} = &\frac{1}{2\sqrt{2}}|110\rangle\Big(\sqrt{0.7}|0\rangle + \sqrt{0.3}|1\rangle\Big) \\ &+\frac{1}{2\sqrt{2}}\Big(|111\rangle + |100\rangle + |101\rangle + |010\rangle \\ &+\ |011\rangle + |000\rangle + |001\rangle\Big)|0\rangle_R.\end{aligned} \tag{36}$$

To load the amplitude $x_1 = \sqrt{0.7}$, the controlled rotation is used to produce

$$\begin{aligned}|\psi_5\rangle_{x_0,x_1} =& \frac{1}{2\sqrt{2}}|110\rangle\Big(\sqrt{0.7}|0\rangle + \sqrt{0.3}|1\rangle\Big) \\ &+ \frac{1}{2\sqrt{2}}|111\rangle\Big[\cos\Big(\frac{\theta_1}{2}\Big)|0\rangle + \sin\Big(\frac{\theta_1}{2}\Big)|1\rangle\Big] \\ &+ \frac{1}{2\sqrt{2}}\sum_{|t\rangle\neq|111\rangle,|110\rangle}|t\rangle|0\rangle_R \end{aligned} \tag{37}$$

With $\theta_1 = 2\arcsin\sqrt{0.7} = 1.1982$, Eq. (37) becomes

$$\begin{aligned}|\psi_5\rangle_{x_0,x_1} =& \frac{1}{2\sqrt{2}}|110\rangle\Big(\sqrt{0.7}|0\rangle + \sqrt{0.3}|1\rangle\Big) \\ &+ \frac{1}{2\sqrt{2}}|111\rangle\Big(\sqrt{0.3}|0\rangle + \sqrt{0.7}|1\rangle\Big) \\ &+ \frac{1}{2\sqrt{2}}\Big(|100\rangle + |101\rangle + |010\rangle \\ &\quad + \ |011\rangle + |000\rangle + |001\rangle\Big)|0\rangle_R. \end{aligned} \tag{38}$$

The second iteration is completed with step 5 of Algorithm 1, yielding a quantum state

$$\begin{aligned}|\psi_6\rangle_{x_0,x_1} =& \frac{1}{2\sqrt{2}}|000\rangle\Big(\sqrt{0.7}|0\rangle + \sqrt{0.3}|1\rangle\Big) \\ &+ \frac{1}{2\sqrt{2}}|001\rangle\Big(\sqrt{0.3}|0\rangle + \sqrt{0.7}|1\rangle\Big) \\ &+ \frac{1}{2\sqrt{2}}\Big(|010\rangle + |011\rangle + |100\rangle \\ &\quad + \ |101\rangle + |110\rangle + |111\rangle\Big)|0\rangle_R. \end{aligned} \tag{39}$$

The efficiency of our storage will be given by the probability $P(|1\rangle)$. We obtain that $P(|1\rangle) = 0.125$, while $P(|0\rangle) = 0.875$. This implies that after the measurement of the state in Eq. (39), the chance of loading x_0 and x_1 in a quantum format as desired is 12.5% until this step of the algorithm.

Due to the probabilistic nature of the FF-QRAM, the post-selection procedure is necessary, which repeats the same algorithm until the register qubit is measured onto $|1\rangle$. Theorem 1 shows that the FF-QRAM can have an exponential computational cost in the worst case.

Theorem 1. *If $|\psi_0\rangle = |+\rangle^{\otimes n}|0\rangle$, $data = \{x_k, p_k\}_{k=0}^{M-1}$, then to create a quantum state $\sum_{k=0}^{M-1} x_k|p_k\rangle$, the FF-QRAM post-selection success probability is $P(|1\rangle) = \frac{1}{2^n}$.*

Proof. According to the FF-QRAM algorithm, the initial state can be written as

$$|\psi_0\rangle = \sum_{p_k\in\mathcal{P}} \frac{1}{\sqrt{2^n}}|p_k\rangle|0\rangle_R + \frac{1}{\sqrt{2^n}}\sum_{t\notin\mathcal{P}}|t\rangle|0\rangle_R,$$

where $\mathcal{P} = \{p_0, \cdots, p_{M-1}\}$. In each iteration of the algorithm, the controlled rotation gate takes $\theta_k = 2\arcsin x_k$ to satisfy Eq. (19). Thus after performing M iterations to load the desired amplitudes, we have the following state.

$$|\psi_f\rangle = \sum_{k=0}^{M-1} \frac{1}{\sqrt{2^n}}|p_k\rangle[y_k|0\rangle + x_k|1\rangle] + \frac{1}{\sqrt{2^n}}\sum_{t\notin\mathcal{P}}|t\rangle|0\rangle_R.$$

Since we want to obtain $QRAM(|\psi_0\rangle) = \sum_{k=0}^{M-1} x_k|p_k\rangle$, we need to study the probability of obtaining $|1\rangle$ when measuring the register qubit. The success probability is calculated as

$$P(|1\rangle) = \sum_{k=0}^{M-1}\frac{1}{2^n}|x_k|^2 = \frac{1}{2^n}\sum_{k=0}^{M-1}|x_k|^2 = \frac{1}{2^n},$$

since $\sum_{k=0}^{M-1}|x_k|^2 = 1$ due to the normalization condition. □

The previous theorem leads to following corollaries.

Corollary 1. *If n, the number of bits, in the quantum state increases, the probability $P(|1\rangle) = \frac{1}{2^n} \to 0$, that is, the chance of success in the process decreases.*

Corollary 2. *If our initial state has exactly the superposition of M computational basis of n qubits, which corresponds to $\mathcal{P}$, that is,*

$$|\psi_0\rangle = \sum_{k=0}^{M-1} \frac{1}{\sqrt{M}} |p_k\rangle|0\rangle,$$

then the probability $P(|1\rangle) = \frac{1}{M} \to 0$ as M increases.

5 Improving FF-QRAM and PQM storing algorithms

In this section, we present methods for reducing the computational cost of the circuit-based quantum random access memory. First, we show that the post-selection probability can be improved by using the PQM algorithm to prepare the input state for the FF-QRAM. For the remainder of this work, the combination of the aforementioned algorithms shall be referred to as FFP-QRAM. Then, we show that the post-selection probability can be further improved by preprocessing the classical data to be loaded. These methods change the computational cost from $O(CMn)$ to $O(C'Mn)$, where $C' < C$. Finally, we present a new loading algorithm inspired by the PQM and FF-QRAM that completes the generalized amplitude encoding without post-selection, reducing the computational cost from $O(CMn)$ to $O(Mn)$.

5.1 FFP-QRAM - Combining PQM and FF-QRAM

The PQM algorithm has computational cost $O(Mn)$ for loading M n-bit binary patterns in superposition with the same amplitude. This is the same computational cost of the FF-QRAM if there is no need to carry out a post-selection.

If we have a quantity M of binary patterns to store in a state with n qubits and $M \ll 2^n$, the PQM algorithm keeps efficient with costs associated with the number of patterns. On the other hand, as shown in the previous section, when trying to store $M \ll 2^n$ patterns, the post-selection process of the FF-QRAM algorithm initialized with $|+\rangle^{\otimes n}$ will have exponential computational cost.

From these two observations, we propose to combine PQM and FF-QRAM algorithms, to store the desired quantity of patterns as a quantum state in superposition.

Consider an input database given by Eq. (1). Suppose we want to obtain a quantum state Eq. (2), with $M \ll 2^n$ terms. We use Algorithm 2 to store the binary patterns p_k. After M iterations of the PQM algorithm, we will have the following final state

$$|\psi_f\rangle_{\text{PQM}} = \frac{1}{\sqrt{M}} \sum_{k=0}^{M-1} |p_k\rangle. \tag{40}$$

This final state Eq. (40) in PQM will be the initial state of the FF-QRAM that will have an increased probability of post-selection $P(|1\rangle)$. However, as described in Corollary 2, $P(|1\rangle)$ approaches 0 when we increase the number of patterns M. To obtain a higher post-selection probability we define a preprocessing procedure in the next section.

5.2 Improvement via Data Preprocessing

The post-selection probability can also be improved with a preprocessing strategy. The preprocessing consists of dividing every entry in the input state by

$$c = \max_{0 \le k < M} (|x_k|).$$

After the preprocessing, Eq. (4) becomes

$$\frac{1}{\sqrt{M}} \sum_{k=0}^{M-1} |k\rangle|p_k\rangle \left(\sqrt{1 - \left|\frac{x_k}{c}\right|^2} |0\rangle + \frac{x_k}{c}|1\rangle \right).$$

After the post-selection measurement we obtain the state

$$\frac{1}{\sqrt{M}} \sum_{k=0}^{M-1} \frac{\frac{x_k}{c}|k\rangle|p_k\rangle|1\rangle}{\sqrt{\frac{1}{M}\sum_{j=0}^{M-1} \left|\frac{x_j}{c}\right|^2}} = \sum_{k=0}^{M-1} x_k|k\rangle|p_k\rangle|1\rangle,$$

and we conclude the state preparation by uncomputing $|k\rangle$. The post-selection probability becomes $\frac{1}{c^2 M}$. We increased the post-selection success probability by a factor $\frac{1}{c^2}$ with $0 < c < 1$. We will demonstrate the impact of the preprocessing strategy in the FF-QRAM and FFP-QRAM with two sets of experiments. Both experiments create a sparse state because this is the worst case for the FF-QRAM.

In the first set of experiments, the input consists of two n-bit patterns with amplitudes $\sqrt{0.3}$ and $\sqrt{0.7}$ and FF-QRAM initial state $|+\rangle^{\otimes n}$. Figure 4 shows the estimated post-selection probability $P(|1\rangle)$ in FF-QRAM for this example with $n = 1, \ldots, 8$, and with and without the preprocessing strategy. There is an improvement in the post-selection probability, but in the worst case the post-selection probability approaches zero and the FF-QRAM will have an exponential computational cost.

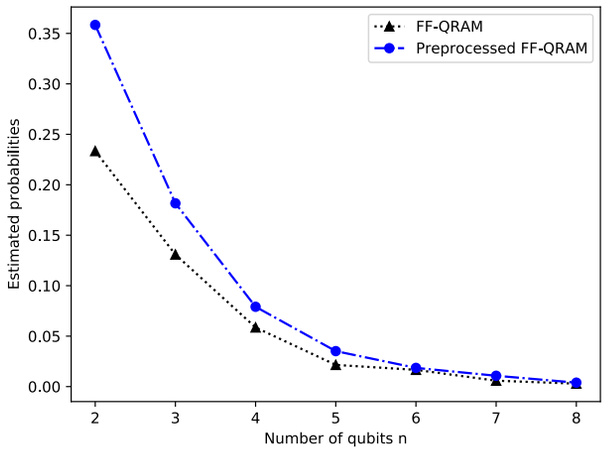


Fig. 4: Comparison between the original FF-QRAM with a FF-QRAM using the preprocessing strategy. The number of zero-valued data entries is $2^n - 2$, where n is the number of qubits represented on the x axis.

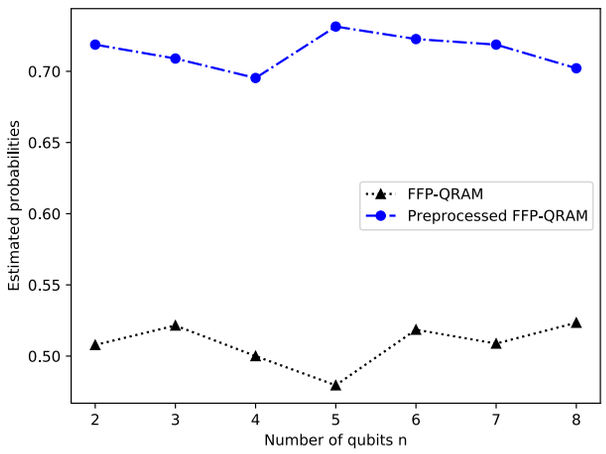


Fig. 5: Comparison between the original FFP-QRAM with a FFP-QRAM using the preprocessing strategy. The number of zero-valued data entries is $2^n - 2$, where n is the number of qubits.

In the second set of experiments, the input consists of two n-bit patterns with amplitudes $\sqrt{0.3}$ and $\sqrt{0.7}$ and we used the FFP-QRAM. Figure 5 shows the estimated post-selection probability $P(|1\rangle)$ in FFP-QRAM for this example with $n = 1, \ldots, 8$, and with and without the preprocessing strategy. The FFP-QRAM has an improved post-selection probability when used to prepare certain sparse vectors and has the same success probability as FF-QRAM when used to prepare a dense state. Given the improvement in the post-selection success probability, in the next sections, we only use the FF-QRAM and FFP-QRAM with the preprocessing strategy.

All quantum circuits were implemented using $python$ programming language, with the quantum computing framework *Qiskit* [19]. To perform the numerical simulations, we used *QASM* simulator within *Qiskit*. The *QASM* simulator is a backend that emulates the implemented quantum circuit as if it was being executed in a real quantum device. To obtain answers in this kind of simulator, it is necessary to perform measurements on the qubits. Since a quantum circuit's output is probabilistic, the simulation must be executed several times. The number of repetitions in all simulations was 1024.

Next, we will present a new algorithm for loading complex data without post-selection.

5.3 A-PQM - Adapted PQM loading procedure for storing continuous data

Based on [13], [14], [20], we propose a modified version of the PQM storage algorithm to load patterns with continuous amplitudes. We modify the S^r operator and adopt the flip flop operators of the FF-QRAM algorithm. The proposed algorithm is capable of storing the desired amount of complex data, such as amplitudes of a state in superposition, with computational cost $O(Mn)$ and without a post-selection procedure. We refer to this algorithm as adapted PQM (A-PQM).

Consider the input database given by Eq. (1). We will use Algorithm 3 to obtain the desired state in Eq. (2), whose initial state, according to our change, will only have $n+2$ qubits, as a consequence of the elimination of the first register $|p\rangle$ of the Algorithm 2 . Thus, the A-PQM storage algorithm has quantum states with structure

$$|u_1 u_2; m\rangle,$$

where the initial state has $u_1 = 0$, $u_2 = 1$ and the memory register $|m\rangle = |m[0]m[1]...m[n-1]\rangle$ initialized in $|0\rangle$.

Under these conditions, using the PQM storage algorithm [14] and the controlled rotations adopted in [13] as bases, our main proposal is a deterministic algorithm, capable of loading complex data without the cost of post-selection. The A-PQM algorithm is described in Algorithm 3, and $|\psi_{k_i}\rangle$ denotes the quantum state in step i of storing the pattern p_k with the amplitude x_k.

Algorithm 3: A-PQM - Complex Data Storage Algorithm

input : data = $\{x_k, p_k\}_{k=0}^{M-1}$

output: $|\psi\rangle = \sum_{k=0}^{M-1} x_k |p_k\rangle$

1 `load` ($data$, $|\psi\rangle$)**:**
2 The initial state $|\psi_{k_0}\rangle = |01; 0, \cdots, 0\rangle$
3 **foreach** $(x_k, p_k) \in data$ **do**
4 **for** $j = 0 \to n-1$ **do**
5 **if** $p_k[j] \neq 0$ **then**
6 $|\psi_{k_1}\rangle = CX_{u_2,m_j}|\psi_{k_0}\rangle$
7 **else**
8 $|\psi_{k_1}\rangle = X_{m_j}|\psi_{k_0}\rangle$
9 $|\psi_{k_2}\rangle = C^n X_{m,u_1}|\psi_{k_1}\rangle$
10 $|\psi_{k_3}\rangle = CU_3^{(x_k,\gamma_k)}{}_{u_1,u_2}|\psi_{k_2}\rangle$
11 $|\psi_{k_4}\rangle = C^n X_{m,u_1}|\psi_{k_3}\rangle$
12 **for** $j = 0 \to n-1$ **do**
13 **if** $p_k[j] \neq 0$ **then**
14 $|\psi_{k_5}\rangle = CX_{u_2,m_j}|\psi_{k_4}\rangle$
15 **else**
16 $|\psi_{k_5}\rangle = X_{m_j}|\psi_{k_4}\rangle$
17 **return** $|\psi\rangle$

The quantum circuit for loading x_k as a complex amplitude of $|p_k\rangle$, is depicted in Figure 6.

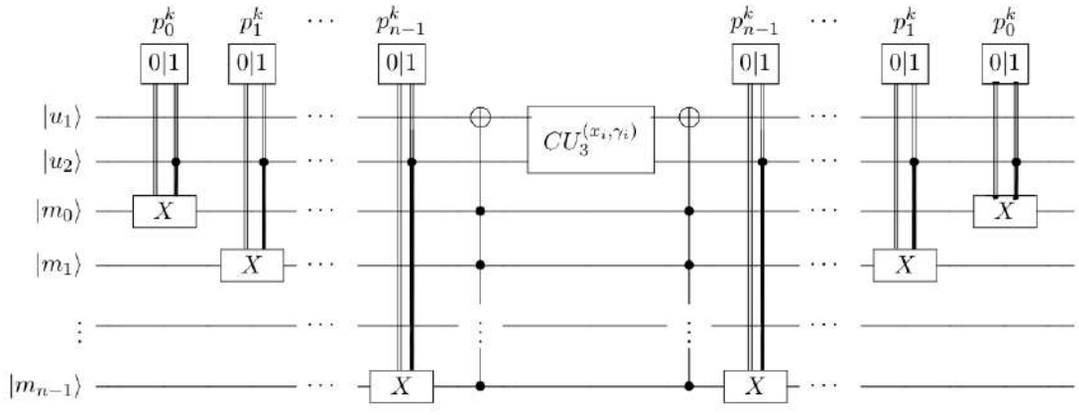


Fig. 6: One iteration of the A-PQM data storage algorithm

Algorithm 3 works in a similar way to the Algorithm 2. That is, steps 4-8 correspond to the loading steps, and steps 12-16 correspond to the unloading steps in Algorithm 2. And given that there are M complex data, each associated with a binary pattern of length n, $O(Mn)$ steps are required to store complex data in Algorithm 3. Thus, we are able to encode complex data in a quantum state in a deterministic way, thereby eliminating the post-selection cost of Algorithm 1. In the next subsection, we will make an example of the quantum data encoding procedure using Algorithm 3.

5.3.1 Example

Let

$$data = \{(\sqrt{0.1} - i\sqrt{0.2}, 00); (\sqrt{0.1} - i\sqrt{0.1}, 01); \\ (\sqrt{0.1}, 10); (\sqrt{0.4}, 11)\},$$

be an input database, determined by a list of four complex numbers that we want to store. The Algorithm 3 will create a state $|\psi\rangle$, as described below.

$$|\psi\rangle = \sum_{k=0}^{3} x_k |p_k\rangle \tag{41}$$

The initial state is given by

$$|\psi_{0_0}\rangle = |01; 00\rangle.$$

For input $(\sqrt{0.1} - i\sqrt{0.2}, 00)$, as $p_0[0]p_1[0] = 00$, by step 4 we have

$$|\psi_{0_1}\rangle = |01; 11\rangle. \tag{42}$$

In step 9, the operator will adjust the auxiliary qubit u_1 to 1, if each qubits in memory $|m[0]m[1]\rangle$ has a value of 1, to produce

$$|\psi_{0_2}\rangle = |11; 11\rangle. \tag{43}$$

Now, in step 10, we have reached the moment where our algorithm differs from the one proposed by [14]. Applying $\gamma_0 = 1$ and $x_0 = \sqrt{0.1} - i\sqrt{0.2}$ in Eq. (10), we get

$$U_3^{(x_0,\gamma_0)} = \begin{bmatrix} \sqrt{0.7} & \sqrt{0.1} - i\sqrt{0.2} \\ -\sqrt{0.1} - i\sqrt{0.2} & \sqrt{0.7} \end{bmatrix}.$$

$|\psi_{0_3}\rangle = CU_3^{(x_0,\gamma_0)}{}_{u_1,u_2}|\psi_{0_2}\rangle$ results in

$$|\psi_{0_3}\rangle = (\sqrt{0.1} - i\sqrt{0.2})|10; 11\rangle + \sqrt{0.7}|11; 11\rangle. \tag{44}$$

Step 11 resets the qubit u_1 to the initial state 0, if each qubit in memory $|m[0]m[1]\rangle$ has a value of 1. Then

$$|\psi_{0_4}\rangle = (\sqrt{0.1} - i\sqrt{0.2})|00; 11\rangle + \sqrt{0.7}|01; 11\rangle. \tag{45}$$

By step 12, we have $|\psi_{0_4}\rangle = X_j|\psi_{0_4}\rangle$. Then

$$|\psi_{0_5}\rangle = (\sqrt{0.1} - i\sqrt{0.2})|00; 00\rangle + \sqrt{0.7}|01; 00\rangle. \tag{46}$$

In Algorithm 3, the qubit u_2 works in the same way as in Algorithm 2. In $|\psi_{0_5}\rangle$, $u_2 = 0$ indicates that the corresponding term stores the input (x_0, p_0) in the memory, while $u_2 = 1$ indicates that the term is being processed to store a new input. After running all iterations of the algorithm, all terms must have $u_2 = 0$ indicating that all patterns are stored, thus ending the algorithm.

Now, for the second input pattern, we have

$$|\psi_{1_0}\rangle = (\sqrt{0.1} - i\sqrt{0.2})|00; 00\rangle + \sqrt{0.7}|01; 00\rangle. \tag{47}$$

For input $(\sqrt{0.1} - i\sqrt{0.1}, 01)$, as $p_1[0] = 0$ the memories $m[0]$ will be changed by the operator X in both terms. As $p_1[1] = 1$, only the memory $m[1]$ of the second term will be changed by the operator $CX_{u_2 m[1]}$, since only in the second term $u_2 = 1$. Then, by proceeding from step 4,

$$|\psi_{1_1}\rangle = (\sqrt{0.1} - i\sqrt{0.2})|00; 10\rangle + \sqrt{0.7}|01; 11\rangle. \tag{48}$$

Following the subsequent steps of the algorithm, we obtain the following state:

$$|\psi_{1_2}\rangle = (\sqrt{0.1} - i\sqrt{0.2})|00; 10\rangle + \sqrt{0.7}|11; 11\rangle.$$

In this iteration, we have $\gamma_1 = 01 - 0.3 = 0.7$. Applying x_1 and γ_1 in Eq. (10), we get:

$$U_3^{(x_1,\gamma_1)} = \begin{bmatrix} \frac{\sqrt{0.5}}{\sqrt{0.7}} & \frac{\sqrt{0.1} - i\sqrt{0.1}}{\sqrt{0.7}} \\ \\ \frac{-\sqrt{0.1} - i\sqrt{0.1}}{\sqrt{0.7}} & \frac{\sqrt{0.5}}{\sqrt{0.7}} \end{bmatrix},$$

and

$$|\psi_{1_3}\rangle = (\sqrt{0.1} - i\sqrt{0.2})|00;10\rangle + (\sqrt{0.1} - i\sqrt{0.1})|10;11\rangle + \sqrt{0.5}|11;11\rangle,$$

$$|\psi_{1_4}\rangle = (\sqrt{0.1} - i\sqrt{0.2})|00;10\rangle + (\sqrt{0.1} - i\sqrt{0.1})|00;11\rangle + \sqrt{0.5}|01;11\rangle,$$

and

$$|\psi_{1_5}\rangle = (\sqrt{0.1} - i\sqrt{0.2})|00;00\rangle + (\sqrt{0.1} - i\sqrt{0.1})|00;01\rangle + \sqrt{0.5}|01;00\rangle.$$

At this moment, in the first two terms of the state $|\psi_{1_5}\rangle$, we have $u_2 = 0$, indicating that the amplitudes x_0 and x_1 are loaded with the states p_0 and p_1, respectively, and the third term with $u_2 = 1$ will be used to load the next input pattern. Therefore, we are ready to proceed to the next entry pattern.

Following the same procedure for the inputs $(\sqrt{0.1}, 10)$ and $(\sqrt{0.4}, 11)$ in the Algorithm 3, we obtain the final states:

$$|\psi_{2_5}\rangle = (\sqrt{0.1} - i\sqrt{0.2})|00;00\rangle + (\sqrt{0.1} - i\sqrt{0.1})|00;01\rangle + \sqrt{0.1}|00;10\rangle + \sqrt{0.4}|01;00\rangle$$

and

$$|\psi_{3_5}\rangle = (\sqrt{0.1} - i\sqrt{0.2})|00;00\rangle + (\sqrt{0.1} - i\sqrt{0.1})|00;01\rangle + \sqrt{0.1}|00;10\rangle + \sqrt{0.4}|00;11\rangle.$$

Note that in $|\psi_{3_5}\rangle$, $u_2 = 0$ in all terms, meaning that there is no more term in the process. Therefore, we successfully prepared the desired state shown in Eq. (41) as

$$|\psi\rangle = (\sqrt{0.1} - i\sqrt{0.2})|00\rangle + (\sqrt{0.1} - i\sqrt{0.1})|01\rangle + \sqrt{0.1}|10\rangle + \sqrt{0.4}|11\rangle. \quad (49)$$

6 Experiments

We performed experiments with the FF-QRAM, FFP-QRAM using the preprocessing strategy presented in Section 5.2, and the adapted version of PQM (A-PQM) for continuous amplitudes. FF-QRAM and FFP-QRAM have the computational cost of $O(CMn)$, and A-PQM has the computational cost of $O(Mn)$. Through experimentation from the initialization of sparse to dense quantum states, we investigate the impact of sparsity in the computational cost of FF-QRAM and FFP-QRAM.

In Figure 7, we verify the probability of post-selection of FF-QRAM, FFP-QRAM, and A-PQM using M 11-bit patterns, with $M = 4, 8, \ldots, 2^{11}$, and amplitudes initialized randomly following a uniform distribution. In this case, the FFP-QRAM has an improved post-selection probability, requiring only a constant number of repetitions to achieve the desired state with high probability, and has computational cost $O(Mn)$.

The post-selection probability of FFP-QRAM depends on the data distribution. In Figure 8, we verify the probability of post-selection of FF-QRAM, FFP-QRAM, and A-PQM using M 11-bit patterns, with $M = 2, 4, 8, \ldots, 2^{11}$, and with an artificial data set in which the largest amplitude is approximately 0.99 and the rest of the data initialized randomly. In this case, the preprocessing will not improve the post-selection probability of FF-QRAM and FFP-QRAM, and they will require an exponential number of repetitions to generate the desired state.

In the worst case, the post-selection success probability of FF-QRAM and FFP-QRAM approaches 0, and it is necessary to perform an exponential number of FF-QRAM or FFP-QRAM calls to prepare a quantum state. The A-PQM requires a polynomial number of operations to initialize a quantum state with M patterns. In the worst case, the A-PQM state preparation is exponentially faster than the FF-QRAM and FFP-QRAM.

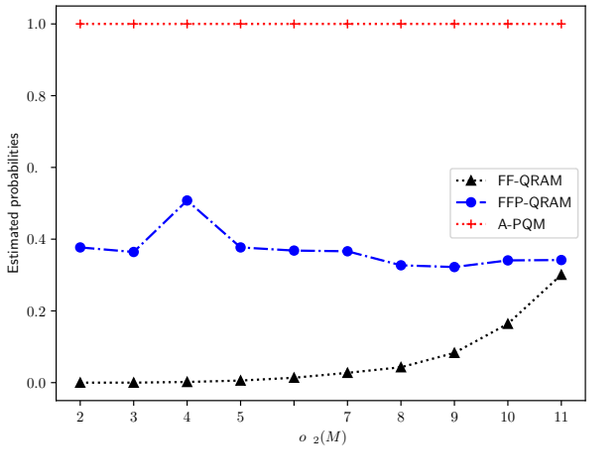


Fig. 7: Success probability to load M patterns into a FF-QRAM, FFP-QRAM or A-PQM with 11 qubits, where the amplitudes are initialized with a random uniform distribution.

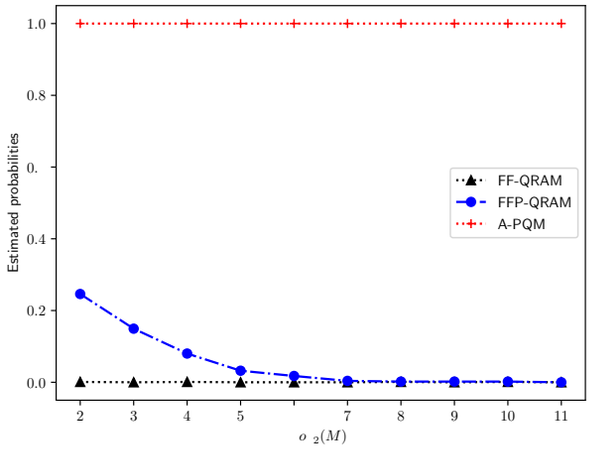


Fig. 8: Success probability to load M patterns into a FF-QRAM, FFP-QRAM or A-PQM with 11 qubits, where the maximum amplitude of the data is about 0.99 and the rest are chosen randomly.

7 Conclusion

The ability to load data in a quantum state efficiently is of critical importance in quantum computing. Ref. [13] proposed a method to load a database structured as M pairs of a complex number and an n-bit pattern in a quantum computer with a computational cost of $O(CMn)$, where C is the number of repetitions for post-selection that depends on the distribution of the data. In this work we showed that C can dominate the computational cost and nullify the efficiency of the algorithm proposed in Ref. [13]. Then we presented several strategies to circumvent this critical issue. We showed that the success probability for post-selection can be improved by combining two known algorithms together and preprocessing the data. Then we presented a new algorithm for loading the quantum database without post-selection, thereby reducing the computational cost to $O(Mn)$. The proposed method is based on the algorithms proposed in Refs. [13] and [14]. We also reduced the number of qubits used in the PQM algorithm from $2n + 2$ to $n + 2$, which is favorable for using the proposed algorithm in noisy intermediate-scale quantum devices.

Acknowledgment

This work was supported by CNPq (Grant No. 308730/ 2018-6), CAPES (Finance code 001), FACEPE (Grant No. BIC-1528-1.03/18), and the National Research Foundation of Korea (Grant No. 2019R1I1A1A01050161).

REFERENCES

[1] Paul Benioff. The computer as a physical system: A microscopic quantum mechanical hamiltonian model of computers as represented by turing machines. Journal of statistical physics, 22(5):563–591, 1980.

[2] Richard P Feynman. Simulating physics with computers. International journal of theoretical physics, 21(6):467–488, 1982.

[3] Seth Lloyd. Universal quantum simulators. Science, 273(5278):1073–1078, 1996.

[4] Lov K Grover. Quantum mechanics helps in searching for a needle in a haystack. Physical review letters, 79(2):325, 1997.

[5] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM review, 41(2):303–332, 1999.

[6] Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd. Quantum support vector machine for big data classification. Physical review letters, 113(13):130503, 2014.

[7] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. Quantum machine learning. Nature, 549(7671):195, 2017.

[8] Gui-Lu Long and Yang Sun. Efficient scheme for initializing a quantum register with an arbitrary superposed state. Physical Review A, 64(1):014303, 2001.

[9] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. Quantum random access memory. Physical Review Letters, 100(16), Apr 2008.

[10] Srinivasan Arunachalam, Vlad Gheorghiu, Tomas Jochym-O'Connor, Michele Mosca, and Priyaa Varshinee Srinivasan. On the robustness of bucket brigade quantum RAM. New Journal of Physics, 17(12):123010, dec 2015.

[11] O. D. Matteo, V. Gheorghiu, and M. Mosca. Fault-tolerant resource estimation of quantum random-access memories. IEEE Transactions on Quantum Engineering, 1:1–13, 2020.

[12] Zhikuan Zhao, Jack K. Fitzsimons, Patrick Rebentrost, Vedran Dunjko, and Joseph F. Fitzsimons. Smooth input preparation for quantum and quantum-inspired machine learning, 2018.

[13] Daniel K Park, Francesco Petruccione, and June-Koo Kevin Rhee. Circuit-based quantum random access memory for classical data. Scientific reports, 9(1):1–8, 2019.

[14] Carlo A Trugenberger. Probabilistic quantum memories. Physical Review Letters, 87(6):067901, 2001.

[15] Michael A Nielsen and Isaac Chuang. Quantum computation and quantum information. AAPT, 2002.

[16] Carlo A Trugenberger. Quantum pattern recognition. Quantum Information Processing, 1(6):471–493, 2002.

[17] Dan Ventura and Tony Martinez. Initializing the amplitude distribution of a quantum state. Foundations of Physics Letters, 12(6):547–559, 1999.

[18] Dan Ventura and Tony Martinez. Quantum associative memory. Information Sciences, 124(1-4):273–296, 2000.

[19] Gadi Aleksandrowicz and et al. Qiskit: An open-source framework for quantum computing, 2019.

[20] Mikko Möttönen, Juha J Vartiainen, Ville Bergholm, and Martti M Salomaa. Transformation of quantum states using uniformly controlled rotations. Quantum Information & Computation, 5(6):467–473, 2005.

4.2 CONTINUOUS VALUED QUANTUM RAM FOR SPARSE STATE PREPARATION

The third algorithm developed in the dissertation research is based on our previous work CV-QRAM (VERAS et al., 2020), designated as CVO-QRAM.

The CVO-QRAM is A deterministic that loads continuous amplitudes at computational cost of $O(Mn)$ steps and classical cost $\mathcal{O}(M \log M + Mn)$ regarding the number of non-zero amplitudes. The CVO-QRAM proved to be more efficient than (PARK; PETRUCCIONE; RHEE, 2019), (VERAS et al., 2020) in all scenarios, in addition to proving, as shown in the experiments, quite competitive in the preparation of sparse quantum states in relation to other state preparation algorithms (SHENDE; BULLOCK; MARKOV, 2006), (MÖTTÖNEN et al., 2005), (PLESCH; BRUKNER, 2011), (ITEN et al., 2016) when $n \gg 1$. Suppose the desired quantum state is doubly sparse. In that case, CVO-QRAM is more efficient than the Pivoting algorithm proposed in (MALVETTI; ITEN; COLBECK, 2021) in preparing a quantum state with $M = 2^s$ non-zero amplitudes for a fixed number of non-zero amplitudes s and $n \gg 1$. The results are even more favorable when $s \gg 1$. Therefore, CVO-QRAM is an efficient quantum state preparation algorithm in preparing sparse quantum states.

Continuous valued quantum RAM for sparse state preparation.

T. M. L. de Veras *[1], L. D. da Silva[1], and Adenilton J. da Silva[2]

[1] *Departamento de Matemática, Universidade Federal Rural de Pernambuco, 52171-900 Recife, PE, Brazil*

[2] *Centro de Informática, Universidade Federal de Pernambuco, 50670-901 Recife, PE, Brazil*

Abstract

Initializing classical data in a quantum device is an essential step in many quantum algorithms. As a consequence of measurement and noisy operations, some algorithms need to reinitialize the prepared state several times during its execution. If the quantum state preparation is not efficient, the quantum state preparation cost can dominate the computational cost of an algorithm. In this work, we propose a quantum state preparation algorithm, called CVO-QRAM algorithm, with computational cost $\mathcal{O}(kM)$, where M is the number of nonzero probability amplitudes and k is the maximum number of bits with value 1 in one of the patterns to be stored. The proposed algorithm can be an alternative to create sparse states in future NISQ devices.

Keywords: Quantum computing, Quantum State Initialization, Optimization of Quantum Algorithms.

1 Introduction

The promise of quantum computing [1, 2, 3, 4] is to solve some problems more efficiently than classical computing. Quantum speed-ups are a consequence of the intrinsic properties of quantum computing [5], [6], as quantum superposition, entanglement, and quantum parallelism. The prime factorization algorithm [7] and the Grovers' search algorithm [8] are examples of quantum algorithms.

Suppose we have a quantum algorithm that receives classical data as input. It will be necessary to prepare a quantum state from the classical data, which will be used as the initialization of the algorithm. Denoting the set of classical data by

$$\mathcal{D} = \left\{ (x_k, p_k) | x_k \in \mathbb{C}, \sum_k |x_k|^2 = 1, p_k \in \{0,1\}^n \right\}, \tag{1}$$

*Corresponding Author: tiago.veras@ufrpe.br

where $0 \leq k < M$. A method to load $\mathcal{D}$ into quantum device is to prepare the state $|\psi\rangle$ with n qubits given by

$$|\psi\rangle = \sum_{k=0}^{M-1} x_k |p_k\rangle. \tag{2}$$

A classical algorithm that creates a circuit $SP_{\mathcal{D}}$, where $SP_{\mathcal{D}}|0\rangle = |\psi\rangle$ is a quantum state preparation algorithm.

Quantum state preparation is an indispensable subroutine in many quantum algorithms [9, 10, 11, 12, 13, 14, 15]. The best quantum state preparation algorithms for dense quantum states have exponential computational cost in relation to the number of qubits, in addition to an exponential computational cost in the classical machine for many of these algorithms.

The quantum Random Access Memory (QRAM) proposed in [16], called bucket brigade (BB-QRAM), is a device capable of storing classical or quantum data, with the ability to consult (read and write) data during quantum information processing in a quantum superposition of states. It requires $O(log_2(M))$ address qubits and $O(M)$ classical or quantum memory cells for M binary data. The Flip-Flop QRAMs is a circuit-based QRAM of computational cost $\mathcal{O}(CnM)$, where C is the number of trials required due to post-selection, n is the number of qubits, and M is the number of patterns.

State preparation algorithms can be classified as exact algorithms [9, 10, 17, 18, 19] and approximated algorithms [11, 20, 21, 22]. This work focuses on the exact state preparation algorithms, which can be grouped in two types: i) algorithms that prepares the quantum states, loading each pattern in a quantum superposition one by one [9], [18], [19] at computational cost related to the number of amplitudes and qubits; ii) algorithms that use decompositions of quantum states to prepare the quantum state by loading all the patterns in a quantum superposition, at exponential computational cost related to the number of qubits in the desired state [23], [24], [25]. In addition, these algorithms have subroutines with associated classical cost, for instance Schmidt Decomposition [25] and decomposition of the quantum state [23].

Algorithms with exponential cost in relation to the number of qubits and input patterns are not efficient and can only be used to generate quantum state with a small number of qubits. The algorithms with computation cost $O(nM)$ require a high number of CNots and are not suitable for NISQ devices.

our main goal is to create an algorithm of computation cost $O(nM)$ with a reduced number of controlled operations. This work is based on the Flip-Flop QRAM (FF-QRAM) [18] and the Continuous Valued Flip-Flop QRAM (CV-QRAM) [19]. It is aimed at optimizing the algorithm proposed in [19], reducing the number of controlled operations required to store each of the input patterns. The optimized CV-QRAM is named CVO-QRAM and have computational cost $\mathcal{O}(kM)$, where k is the max number of 1s in the binary patterns and M is the number of patterns.

The main difference between the CVO-QRAM and the FF-QRAMs is the lower number of control qubits in the flip-flop operation. One of the factors that have limited the efficiency of many quantum state preparation algorithms [26, 18, 23, 19] is the large number of controlled operations required for the algorithm to perform its task. In FF-QRAM and CV-QRAM [26, 18, 19], each iteration requires the application of an operation controlled by n qubits

with $\mathcal{O}(n)$ cost,The noise caused by the excess of controlled gates has a negative effect on the generated quantum states.

In this context, this study introduces the quantum state preparation algorithm that optimizes the CV-QRAM algorithm, called Continuous Valued Flip-Flop QRAM Optimization (CVO-QRAM). The CVO-QRAM showed better results in all states preparation scenarios (dense or sparse) in relation to other preparation algorithms [26], [19],[18] that store the input patterns in memory one by one at a cost depending on the number of qubits and the number of input patterns.

Comparing with state preparation algorithms [23, 17, 25] whose cost only depends on the number of qubits, we demonstrate that that CVO-QRAM can be recommended if we wish to prepare a sparse quantum state.Recently, [27] has proposed a sparse quantum state preparation algorithm. The CVO-QRAM requires a lower number of CNOTS in the double sparse case (sparse in relation to the number of amplitudes and number of 1s in p_k)

1.1 Quantum Operators

This section defines some of the quantum operators that will be used in this work. All operators that are not mentioned here can be easily found at [5].

Given a quantum state $|\psi\rangle$ containing bits a and b, we use

$$CX_{(a,b)}|\psi\rangle \tag{3}$$

to indicate a controlled-X operation, where the control and the target are indicated by a subscript a and b, respectively. When this operator is applied in the quantum state $|\psi\rangle$, with control in a, it rewrites the target qubit $|b\rangle$ to $|b \oplus a\rangle$.

According to [5], any quantum operator that acts on a single qubit can be represented up to a global phase by the following matrix:

$$U = \begin{bmatrix} \cos\frac{\theta}{2} & -e^{i\lambda}\sin\frac{\theta}{2} \\ \\ e^{i\phi}\sin\frac{\theta}{2} & e^{i(\lambda+\phi)}\cos\frac{\theta}{2} \end{bmatrix}, \tag{4}$$

As show in [19], it is possible to obtain θ, λ, and ϕ, such that the unitary operator U can be rewritten as a matrix that depends on parameters, x_k and γ_k, as demonstrated in Eq. (5)

$$U^{(x_k,\gamma_k)} = \begin{bmatrix} \sqrt{\frac{\gamma_k - |x_k|^2}{\gamma_k}} & \frac{x_k}{\sqrt{\gamma_k}} \\ \frac{-x_k^*}{\sqrt{\gamma_k}} & \sqrt{\frac{\gamma_k - |x_k|^2}{\gamma_k}} \end{bmatrix}. \tag{5}$$

In CV-QRAM [19], the value of $\gamma_k = \gamma_{k-1} - |x_{k-1}|^2$ is a complex iteration variable, with $1 \leqslant k \leqslant M-1$, and initial condition $\gamma_0 = 1$. Furthermore, $x_k = \alpha + i\beta$ is a complex number whose complex conjugate is denoted by x_k^*.

Now, the operators given by Eq. (3) and Eq.(5) allow us to define the t-controlled operator that will be used in this work. Given a quantum state $|\psi\rangle$ with $n+1$ qubits, which contains the $t+1$ qubits $a_0, a_1, \ldots, a_{t-1}, b$ with $t \leq n$, we employ

$$C^t U^{(x_k,\gamma_k)}{}_{(a_0 a_1 \ldots a_{t-1}, b)}|\psi\rangle \tag{6}$$

to denote a t-controlled operator, where the t qubits $(a_0, a_1, \ldots, a_{t-1})$ denote the t operation controls, and the b denotes the target qubit, to which we will apply the $U^{(x_k,\gamma_k)}$ operator if all the controls have a value of 1 simultaneously. The controlled operation described in Eq. (6) is responsible for loading the complex number x_k as an amplitude associated with the p_k pattern in a term of the desired quantum state.

1.1.1 Decomposition of multi-controlled quantum gates

In current systems, n-qubits controlled gates C^nU are not available as a native machine instruction. Since any quantum gate can be implemented using single-qubit gates and CNOT quantum operator [28], we adopted a decomposition for an operator of the type C^nU combining the decompositions proposed in [5], [28].

According to [5], the quantum gate C^nU can be deposed into $2(n-1)$ Toffoli gates and one single quantum gate CU, using $n-1$ auxiliary qubits prepared in $|0\rangle$. For example, the Figure 1 shows the decomposition of a C^3U gate as proposed in [5].

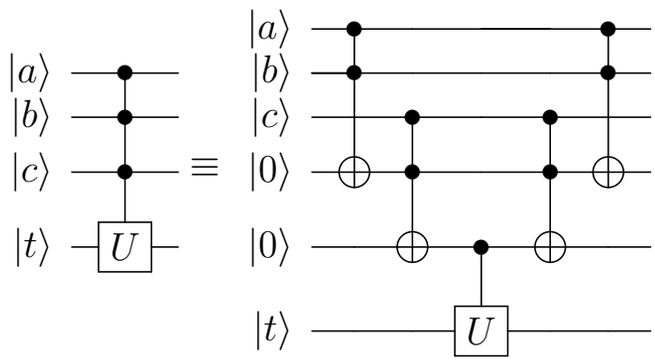


Figure 1: Decomposition of a C^3U as proposed in the [5]

Toffoli gates generated by decomposition [5] are applied in pairs. Then, we apply the decomposition proposed in [28], where 3 CNOT gates are required to implement a Toffoli gate. Lastly, we can implement the CU gate using 2 CNOT gates. Therefore, we can implement a C^nU operator using $6n-4$ CNOT gates.

1.2 Related Works

The Probabilistic Quantum Memory (PQM) state preparation algorithm [26] receives a set of binary patterns $data = \cup_{k=0}^{M-1}\{p_k\}$, where $p_k \in \{0,1\}^n$ and stores them with cost $O(nM)$ in a quantum state in superposition with uniform amplitudes given by

$$|\psi\rangle = \frac{1}{\sqrt{M}} \sum_{k=0}^{M-1} |p_k\rangle. \tag{7}$$

In order to store the M binary patterns with n bits, the algorithm has a computational cost (total operations) $O(nM)$ and requires $14n-2$ CNOT gates to load each binary pattern into the memory register. The Flip-Flop Quantum RAM (FF-QRAM) is a probabilistic quantum

state preparation algorithm that receives a set of structured data given by Eq. (1) and creates a specific quantum input state for a quantum algorithm, given by the superposition of states of the computational basis, with n-qubits, represented by Eq. (2), whose amplitudes are not necessarily equal.

Thus, the FF-QRAM algorithm requires at least $6n - 4$ CNOT gates for each pattern at computational cost $\mathcal{O}(CnM)$, where C is the number of required trials due to post-selection used in the FF-QRAM, n is the number of qubits, and M is the number of patterns [18]. In the worst case, the number of FF-QRAM trials is exponential [19].

The CV-QRAM [19] is a deterministic algorithm based on [26, 18] capable of storing complex data as continuous probability amplitudes and does not require post-selection.The CV-QRAM receives as input a data a set with M input patterns, given by Eq.(1), and outputs a superposition quantum states prepared given by Eq.(2) at a cost of $O(Mn)$ steps.

1.3 Work structure

The remainder of this paper is structured as follows: Section 2 presents the CV-QRAM algorithm [19]. Section 3 describes the proposed CVO-QRAM algorithm. Section 4 presents the results of the experiments performed and shows the improvements achieved by the proposed algorithm. Finally, Section 5 refers to the conclusion.

2 CV-QRAM Algorithm

The CV-QRAM [19] receives as input a data set (Eq. (1)) and provides as output a quantum circuit $SP_{\mathcal{D}}$ with $SP_{\mathcal{D}}|0\rangle = |\psi\rangle$, as described in Eq. (2). CV-QRAM is a deterministic algorithm that depends on the number of input patterns M and the number of n qubits. It performs an exact preparation of the desired quantum state, with complex amplitudes, eliminating the post-selection at computational cost $\mathcal{O}(Mn)$ steps to store M input patterns with n qubits.

Using two quantum registers, the structure of the quantum state in CV-QRAM is $|u; m\rangle$, where $|u\rangle = |u_1 u_2\rangle$ is a auxiliary register, initialized in $|01\rangle$, and $|m\rangle^{\otimes n}$ is a memory register, initialized in $|0\rangle^{\otimes n}$. The quantum circuit corresponding to a single iteration of the CV-QRAM, responsible for loading an input pattern (x_k, p_k) in the quantum state being prepared, as in Figure 2.

During the iteration to store the first pattern, our quantum state has a single term, and the moment the $CU^{(x_k,\gamma_k)}$ rotation gate is applied, where $U^{(x_k,\gamma_k)}$ is described in Eq. (5), the auxiliary register has a value of $|u\rangle = |11\rangle$. After this procedure, an overlap of states is created in the auxiliary qubit $|u_2\rangle$, and the quantum state will have two terms.

By the end of the iteration, we will have $|u_2\rangle = |0\rangle$ in the first term, indicating that the procedure for storing the input pattern (x_k, p_k) has been completed since p_k is loaded into the memory, with amplitude x_k associated. In the second term, we have $|u_2\rangle = |1\rangle$, indicating that this term is still in process and ready to receive the next input.

Therefore, the auxiliary qubit $|u_2\rangle$ divides the quantum state into two parts: one containing the terms where the patterns are already stored, and the other containing the term that is being

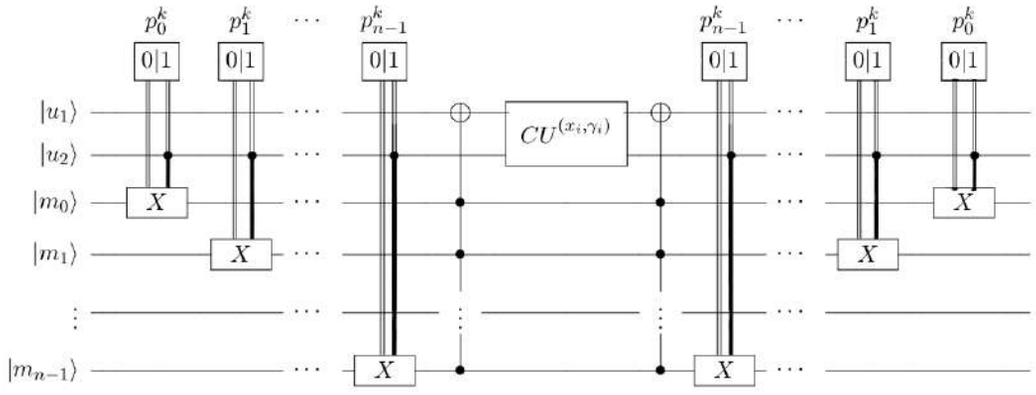


Figure 2: One iteration of the CV-QRAM algorithm to store a (x_k, p_k) input pattern.

processed to receive a new pattern. In the end of the last interaction, we will have $|u_2\rangle = |0\rangle$ in all terms, indicating that the algorithm has performed its task. The CV-QRAM pseudo-code is described in Algorithm 1, where $|\psi_{k_i}\rangle$ denotes the quantum state in step i to store a (x_k, p_k) input pattern.

We can describe the CV-QRAM procedure (Algorithm 1) to prepare a quantum state from input data set $\mathcal{D}$, as follows:

Step 1 loads the data input $\mathcal{D}$ and the Step 2 produces the following initial state:

$$|\psi_{0_0}\rangle = |01; 0, \cdots, 0\rangle,$$

where $|u_1u_2\rangle = |01\rangle$ and $|m\rangle = |0\rangle^{\otimes n}$ are the initial values of the registers. Through the Step 3, for each $(x_k, p_k) \in \mathcal{D}$, the following procedures will be performed: for each $j = 0, \ldots n-1$, providing:

- If the qubit in the position $p_k[j] \neq 0$, write $|\psi_{k_1}\rangle = CX_{u_2,m_j}|\psi_{k_0}\rangle$.
- Others, write $|\psi_{k_1}\rangle = X_{m_j}|\psi_{k_0}\rangle$

After assessing all j values, we obtain the quantum state $|\psi_{k_1}\rangle$. In Step 9, the next quantum state is obtained through the $|\psi_{k_2}\rangle = C^n X_{m,u_1}|\psi_{k_1}\rangle$ operation, which is controlled by all memory qubits, changing the state of the auxiliary qubit u_1. However, after performing the Step 9, a single term is generated in $|\psi_{k_2}\rangle$, where the auxiliary qubits will simultaneously have their values set to 1.

Through the Step 10, the next state is given by:

$$|\psi_{k_3}\rangle = CU^{(x_k,\gamma_k)}{}_{u_1,u_2}|\psi_{k_2}\rangle \tag{8}$$

This operation will be performed precisely in the term where the memory register is $|u\rangle = |11\rangle$. When the rotation gate $U^{(x_k,\gamma_k)}$ is applied to the auxiliary qubit $|u_2\rangle$, an overlay will be created, generating a new term.

The Step 11, from which we obtain the state

$$|\psi_{k_4}\rangle = C^n X_{m,u_1}|\psi_{k_3}\rangle \tag{9}$$

Algorithm 1: CV-QRAM - Complex Data Storage Algorithm

input : data $= \{x_k, p_k\}_{k=0}^{M-1}$

output: $|\psi\rangle = \sum_{k=0}^{M-1} x_k |p_k\rangle$

`load` $(data, |\psi\rangle)$ **:**

- The initial state $|\psi_{0_0}\rangle = |01; 0 \dots 0\rangle$
- **foreach** $(x_k, p_k) \in data$ **do**
  - **for** $j = 0 \to n-1$ **do**
    - **if** $p_k[j] \neq 0$ **then**
      - $|\psi_{k_1}\rangle = CX_{u_2,m_j} |\psi_{k_0}\rangle$
    - **else**
      - $|\psi_{k_1}\rangle = X_{m_j} |\psi_{k_0}\rangle$
  - $|\psi_{k_2}\rangle = C^n X_{m,u_1} |\psi_{k_1}\rangle$
  - $|\psi_{k_3}\rangle = CU^{(x_k,\gamma_k)}{}_{u_1,u_2} |\psi_{k_2}\rangle$
  - $|\psi_{k_4}\rangle = C^n X_{m,u_1} |\psi_{k_3}\rangle$
  - **for** $j = 0 \to n-1$ **do**
    - **if** $p_k[j] \neq 0$ **then**
      - $|\psi_{k_5}\rangle = CX_{u_2,m_j} |\psi_{k_4}\rangle$
    - **else**
      - $|\psi_{k_5}\rangle = X_{m_j} |\psi_{k_4}\rangle$
- **return** $|\psi\rangle$

reverses the operation performed in Step 9, while Step 12, which provides the quantum state $|\psi_{k_5}\rangle$, reverses the operation performed in Step 3.

When the iteration is complete, the term that was being processed will have $|u_2\rangle = |0\rangle$, indicating that the process of preparing the input pattern (x_k, p_k) is complete. While the new term obtained after applying the rotation gate will have $|u_2\rangle = |1\rangle$, indicating that the term is in process and ready to receive the next pattern (x_{k+1}, p_{k+1}).

Particularly in the last interaction of the algorithm, when the (x_{M-1}, p_{M-1}) input pattern is being stored, the application of the operator $CU_{u_1,u_2}^{(x_{M-1},\gamma_{M-1})}$ will not create a new term in the quantum state. In this case, only the auxiliary qubit will be set to zero. After M iterations, we will have $|u_2\rangle = |0\rangle$ in all terms of the quantum state, indicating that the CV-QRAM algorithm has completed its task. Therefore, the desired quantum state $|\psi\rangle$ is prepared from $\mathcal{D}$.

3 CV-QRAM Optimization Algorithm

The CV-QRAM eliminates the post-selection contained in the FF-QRAM algorithm. However, it uses a large number of controlled operations to perform its task. In contrast, the number of total controlled operations required by the CV-QRAM algorithm depends on the number of input patterns M in the data set $\mathcal{D}$ and number n of qubits. Therefore, to store a pattern of type (x_k, p_k), the cost of the algorithm is always the same, regardless of the pattern having all bits equal to zero or one. In addition, we also noticed that the CV-QRAM uses two ancilla bits; one of them is used only for the rotation operator given by the matrix Eq. (5) to be applied.

This section presents an optimization proposal for CV-QRAM algorithm, designated as CVO-QRAM algorithm. The CVO-QRAM is a quantum state preparation algorithm that receives an input data set given by the Eq. (1) and outputs a desired quantum state given by Eq. (2).

In CVO-QRAM, the amount of controlled operations are determined by the number of bits with a value of 1 in the patterns and the position where they occur. Our proposal will demonstrate that this will reduce the number of necessary controlled operations. For instance, in the best scenario, iterating to store a pattern where all bits are zero, no controlled operations are necessary.

3.1 Establishing the partial order of storage

The algorithm requires that the storage order of the patterns provided by the input data set is established by the number of bits with a value of 1 in the binary string. Thus, the patterns must be ordered so that the number of bits with a value of 1 in the binary string is increased. In the CVO-QRAM algorithm, we denote by $p_k[i]$ the i-th bit from a p_k pattern.

If (x_r, p_r) input pattern is stored in the quantum state before (x_s, p_s) input pattern due to $||p_r||_2 \leqslant ||p_s||_2$, where $||p_k||_2$ denotes the Euclidean norm of the given by

$$||p_k||_2 = \sqrt{\sum_{i=0}^{M-1} p_k^2[i]}.$$

3.2 Defining controlled operations according to storage patterns

To prepare for a quantum state with n qubits, the PQM and CV-QRAM algorithms require at least one C^nU operation for each iteration performed to load an input pattern. To load an input pattern (x_k, p_k) into a quantum state, the controlled operations are established by the number of 1s and the position where these bits with a value of 1 occur in the string of the pattern p_k.

If t denotes the number of bits with value 1 in the binary string of a pattern p_k, we reach a total of $2t$ CX operations and one t-controlled operation C^tU, arranged as follows:

a) For each j, where $p_k[j] = 1$, one operation $CX_{(u,p_k[j])}$, generating a total of t operations of this type.

b) One operation $C^tU^{(x_k,\gamma_k)}$ with control in all t bits where $p_k[j] = 1$ and target in $|u\rangle$.

c) . For each j, where $p_k[j] = 1$, one operation of type $CX_{(u,p_k[j])}$ will be carried out, undoing the first t carried out.

For instance, Figures 3 and 4 illustrate the circuits of the CV-QRAM and CVO-QRAM algorithms to store an input pattern, where $|p_k\rangle = |001\rangle$, with a single qubit of value of 1, which occur in position $p_k[2]$.

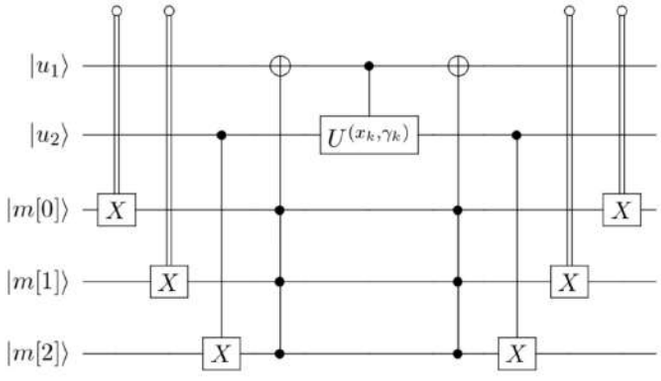


Figure 3: Circuit using CV-QRAM.

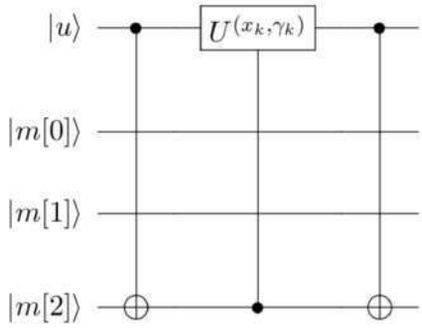


Figure 4: Circuit using CVO-QRAM.

The CV-QRAM algorithm needs 18 controlled operations. In contrast, CVO-QRAM needs 4 controlled operations, showing significantly less CNOT gates required by CVO-QRAM than the CV-QRAM when our binary string has few 1s.

The $U^{(x_k,\gamma_k)}$ rotation operator is responsible for loading the x_k amplitude in the p_k pattern in the end of each algorithm iteration. Particularly, if all bits in the pattern are zero, then the U operator will be applied directly in the $|u\rangle$ qubit without control. Another peculiarity happens when storing the last entry pattern. In this case, the last sequence of operations CX does not need to be performed. Therefore, the CVO-QRAM cost is related with the number of patterns p_k, and the number of $1s$ per $p_k \in \mathcal{D}$.

The auxiliary register $|u\rangle$works in a similar way to the auxiliary qubit $|u_2\rangle$ of CV-QRAM, it keeps dividing the quantum state into terms where the patterns are stored and the term in

processing, indicated by $|u\rangle = |0\rangle$ and $|u\rangle = |1\rangle$, respectively. At each iteration of the CVO-QRAM algorithm, the $U^{(x_k,\gamma_k)}$ operator, responsible for loading a p_k pattern, with associated amplitude x_k, is applied to the $|u\rangle$ qubit of the term being processed. After all the iterations are completed, the auxiliary record of all terms is set to $|u\rangle = |0\rangle$, indicating that the algorithm has completed its work.

Consider that (x_k, p_k) is the input pattern to be stored, t the number of bits with value 1 in the string binary and l as a list containing the positions where $p_k[j] = 1$. If $t = 0$, no controlled operation is necessary, therefore, we only apply the $U^{(x_k,\gamma_k)}$ operator to the auxiliary qubit $|u\rangle$. If $t \geqslant 1$ according to the configuration of the p_k pattern, we have $2t + 1$ controlled operations that will be performed in quantum state $|\psi\rangle$, as follows:

$$QC^t U_{(m[l_0,l_1,\ldots l_t],u)} Q^\dagger |\psi\rangle, \tag{10}$$

where $l_0, l_1, \ldots, l_t \in l$ and Q is the quantum operator, described below:

$$Q \;=\; CX_{(u,m[l_t])} \cdots CX_{(u,m[l_1])} \cdot CX_{(u,m[l_0])}.$$

When preparing a quantum state with n qubits, the most expensive case for CVO-QRAM occurs in the iteration of the pattern where $t = n$. In this case, $2n + 1$ controlled operations will be carried out. Figure 5 illustrates the circuit representing this algorithm iteration.

The decomposition of C^nU gates into CNOT gates proposed in this work will require $8n-4$ CNOT gates to store the pattern, whose configuration establishes the circuit described in the Figure 5.

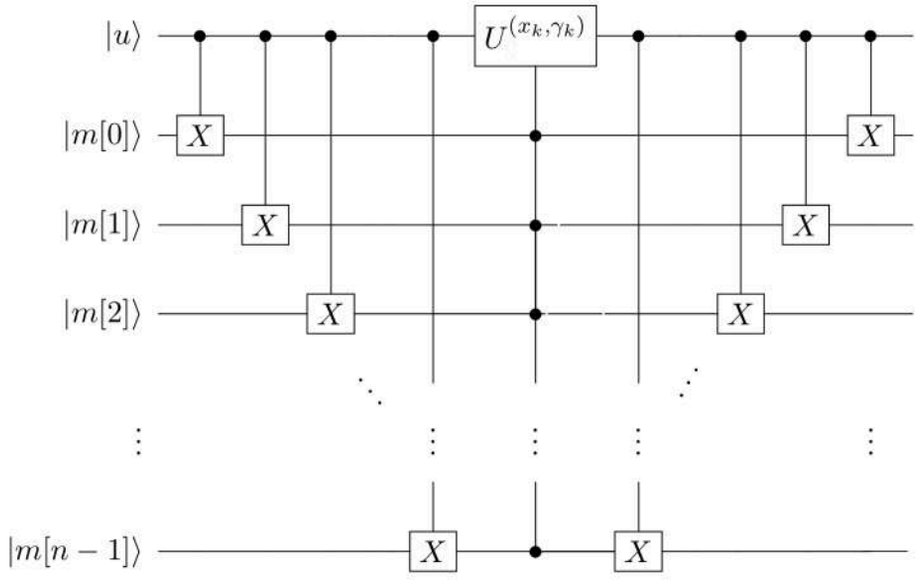


Figure 5: Circuit to store the standard p_k with n qubits, where all qubits have a value of 1, using CVO-QRAM.

3.3 CVO-QRAM Algorithm

CVO-QRAM is the quantum state preparation algorithm proposed in this work, whose pseudo-code can be verified in Algorithm 2. In Algorithm 2, the index s in the state $|\psi_{k_s}\rangle$ denotes the step s of the state preparation of the input pattern (x_k, p_k), where x_k is loaded as amplitude of the binary pattern p_k in one term of the quantum state.

Algorithm 2: CVO-QRAM - Complex Data Storage Algorithm

input : data $= \{x_k, p_k\}_{k=0}^{M-1}$

output: $|\psi\rangle = \sum_{k=0}^{M-1} x_k |p_k\rangle$

1 `load` $(data, |\psi\rangle)$**:**

2 The initial state $|\psi_{0_0}\rangle = |1; 0, \cdots, 0\rangle$

3 **foreach** $(x_k, p_k) \in data$ **do**

4 $t =$ The number of bits with value 1 in the pattern p_k

5 $l =$ a list containing the positions of p_k where $p_k[j] = 1$.

6 $|\psi_{k_1}\rangle = \prod_{l_i \in l} CX_{(u, m[l_i])} |\psi_{k_0}\rangle$

7 $|\psi_{k_2}\rangle = C^t U_{(m[l_0, l_1, \ldots l_t], u)} |\psi_{k_1}\rangle$

8 $|\psi_{k_3}\rangle = \prod_{l_i \in l} CX_{(u, m[i])} |\psi_{k_2}\rangle$

9 **return** $|\psi\rangle$

Function `load` in line 1 of Algorithm 2 prepares the quantum state described by $\mathcal{D}$. Step 2 produces the following initial state:

$$|\psi_{0_0}\rangle = |1; 0, \cdots, 0\rangle,$$

where $|u\rangle = |1\rangle$ and $|m\rangle = |0\rangle^{\otimes n}$ are the initial values of the registers. In Step 3, for each $(x_k, p_k) \in \mathcal{D}$, the variable t denotes the number of bits with value 1 in the binary pattern p_k, while the variable l denotes a list containing the values j, such that $p_k[j] = 1$ in p_k. The Step 6 given by

$$|\psi_{k_1}\rangle = \prod_{j \in l} CX_{(u, m[j])} |\psi_{k_0}\rangle. \tag{11}$$

causes for each $j \in l$, one controlled operation is determined of the type $CX_{(u, m[j])}$, with control in the auxiliary qubit $|u\rangle$ and target in $m[j]$, leading the pattern p_k to be loaded into the memory of the term that is being processed, after t CNOT operations are performed sequentially in state $|\psi_{k_0}\rangle$. The Step 7 given by

$$|\psi_{k_2}\rangle = C^t U^{(x_k, \gamma_k)}{}_{\left(\prod_{j \in l} m[j], u\right)} |\psi_{k_1}\rangle \tag{12}$$

will perform a t-controlled operation by applying $U^{(x_k, \gamma_k)}$ to $|u\rangle$ qubit, controlled by all $p_k[j]$ qubits, where $j \in l$. The $U^{(x_k, \gamma_k)}$ operator will be applied in the only term that is being processed of the $|\psi_{k_1}\rangle$ state.

After being applied, the rotation operator will create a superposition of states in the auxiliary qubit $|u\rangle$, adding one more term to the state. The first term is the one that was in processing and will have its auxiliary qubit value set to $|u\rangle = |0\rangle$, indicating that the x_k amplitude was loaded in this term of state. The second term created after applying the operator will generate $|u\rangle = |1\rangle$, indicating that this term is in processing and the next input pattern (x_{k+1}, p_{k+1}) will be prepared in it.

Finally, step 8 given by

$$|\psi_{k_3}\rangle = \prod_{j \in l} CX_{(u,m[j])}|\psi_{k_2}\rangle, \tag{13}$$

will reverse the t CNOT operations performed on Eq. (6). After applying Eq. (13), we will have in the $|m\rangle$ memory of the terms where $|u\rangle|0\rangle$, the binary patterns p_k stored, while in the term where $|u\rangle = |1\rangle$, the memory is reset, that is, $|m\rangle = |0\rangle$.

Particularly in the last iteration of the Algorithm 2, after applying the $U^{(x_{M-1},\gamma_{M-1})}$ rotation operator, a superposition of states will not be created in the auxiliary qubit of the term in processing, since there will be no more input patterns to be loaded. Therefore, in this last iteration, in all terms of the quantum state, we will have in the auxiliary qubit $|u\rangle = |0\rangle$, indicating that all input patterns (x_k, p_k) are prepared in the state. In addition, the operation performed by Eq. (13) does not need to be performed because of $|u\rangle = |0\rangle$. Thus, the last p_{M-1} pattern is stored and no memory needs to be reset. After M iterations of the Algorithm 2, based on $\mathcal{D}$ input data, we will have prepared the quantum state desired in Eq.(2).

3.4 CVO-QRAM Algorithm cost

Implementing quantum gates that act simultaneously on three or more qubits is a difficult task in real quantum devices. This is currently possible by decomposing these multi-controlled gates into gates that act on one or two qubits [28].

CNOT gates are currently the only 2-qubits gates implemented in quantum computer hardware, whose implementation makes the results of the algorithms more susceptible to errors than gates with one qubit [17]. Thus,CNOT gates generates greater noise than the gate that acts on one qubit.

The amount of CNOT gates required for the algorithm to perform its task is one of the methods to assess the cost of the algorithm [23], [17], [25]. Therefore, we will employ the number of CNOT gates to measure the cost to the CVO-QRAM algorithm to perform its task.

3.4.1 Classical cost CVO-QRAM

Given an input pattern $\mathcal{D}$ with M patterns of (x_k, p_k) where x_k are complex amplitudes and p_k are binary patterns of n bits, the CVO-QRAM has the following classical costs:

- To sort the M input patterns, the CVO-QRAM has a classical cost of $\mathcal{O}(M \log M)$.
- For each iteration k of the CVO-QRAM algorithm, a quantum operator $C^t U^{(x_k,\gamma_k)}$ is applied to the quantum state $|\psi_{k_1}\rangle$. This operator is responsible for loading the amplitude x_k to the pattern $|p_k\rangle$ in the term being processed, where the operator $U^{(x_k,\gamma_k)}$ depends

on the variables (x_k, γ_k), where $\gamma_k = \gamma_{k-1} - |x_{k-1}|^2$ has initial condition $\gamma_0 = 1$, in CVO-QRAM this classical cost is $\mathcal{O}(M)$ to calculate the matrices.

- For each pattern p_k, the algorithm adds $O(n)$ gates in the quantum circuit with a total of $O(nM)$ steps to create the circuit.

Then, $O(M \log M + nM)$ is the overall cost of the CVO-QRAM algorithm on the classical device.

3.4.2 Quantum cost CVO-QRAM

This section is aimed at clarifying the quantum cost of the CVO-QRAM algorithm based on the number of CNOT gates needed to store an input of the type (x_k, p_k), where $p_k \in \{0,1\}^n$. **Cost of the algorithm for dense quantum states.** To prepare quantum state with $M = 2^n$ inputs of n qubits, the number of CNOT gates can be obtained by the following function:

$$f_{\mathcal{D}}(n) = \sum_{t=1}^{n} C(n,t)(8t-4) - n, \tag{14}$$

where $f_{\mathcal{D}}(n)$ denotes the cost function of the number of CNOT gates needed to prepare a dense quantum state with n qubits and denotes $C(n,t)$ as the binomial coefficient.

Cost of the algorithm for sparse quantum states. To prepare a sparse quantum state with $M \leq 2^n$ inputs of n qubits, the required number of CNOT gates can be achieved through the following function:

$$e_{\mathcal{D}}(n) = \sum_{t=1}^{n} \mu_t(8t-4) - t_{\max}, \tag{15}$$

where $e_{\mathcal{D}}(n)$ denotes the cost function of the number of CNOT gates required to prepare a sparse quantum state with n qubits, μ_t is the number of input patterns p_k in $\mathcal{D}$ with t bits with value 1 in the binary string and $t_{\max}$ denotes the highest value of t obtained between patterns p_k.

The best scenario for the CVO-QRAM algorithm to perform its task occurs when the patterns p_k in $\mathcal{D}$ have the smallest number of 1s in the binary string as possible, in accordance with the states of the base. This is equivalent to the set $\mathcal{D}$ being double sparse. In turn, the worst scenario for the CVO-QRAM algorithm occurs when the total number of bits with a value of 1 summed all 1s in all patterns is close to 50%. In both cases, we will find that the number of controlled operations required by CVO-QRAM to load the (x_k, p_k) pattern is less than CV-QRAM.

In the sequence, we exemplify in detail the functioning of the Algorithm 2 based on the input data provided.

3.5 Simple example

Consider that

$$\begin{aligned} data = &\{(\sqrt{0.1} - i\sqrt{0.2}, 00); (\sqrt{0.1}, 10); (\sqrt{0.1} - i\sqrt{0.1}, 01); \\ &(\sqrt{0.4}, 11)\}, \end{aligned}$$

is an input database determined by a list of four complex numbers x_k that we wish to load in patterns p_k. The use of the Algorithm 2 creates a state $|\psi\rangle$, as described below.

$$|\psi\rangle = \sum_{k=0}^{3} x_k |p_k\rangle \tag{16}$$

From the database, we can establish a storage sequence determined by the number of bits with value of 1 appearing in each pattern. Therefore, we will start with the pattern $p_0 = 00$, which does not contain the bit 1 in the pattern. Then, there are two patterns with a bit with a value of 1 that we can store in any order. We adopted the storage orders of $p_1 = 10$ and $p_2 = 01$. Finally, consider storing the state $p_3 = 11$, which is the last and only pattern with two bits with value of 1.

The circuit for this procedure is given below:

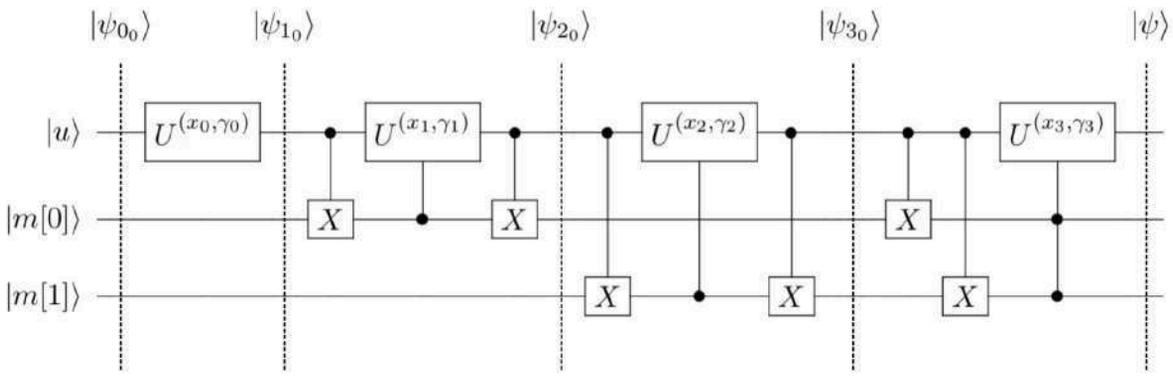


Figure 6: Circuit to load 4 complex amplitudes in 2-bit patterns.

The initial state is given by:

$$|\psi_{0_0}\rangle = |1; 00\rangle.$$

We have a single term: the auxiliary qubit $|u\rangle = |1\rangle$. It indicates that in this term, the input pattern (x_0, p_0) is prepared in the quantum state, where $x_0 = \sqrt{0.1} - i\sqrt{0.2}$ is loaded as amplitude of pattern $p_0 = 00$.

According to the Algorithm 2, as $p_0 = 00$, we obtain $t = 0$ and l as a empty list. Thus, we obtain $|\psi_{0_1}\rangle = |\psi_{0_0}\rangle$. We shall proceed to the Step 7, where we will apply the $U^{(x_0,\gamma_0)}$ operator rotation in the auxiliary qubit $|u\rangle$, without using controls, as seen on the circuit in Figure 6. Applying $x_0 = \sqrt{0.1} - i\sqrt{0.2}$ in Eq.(5), with initial condition $\gamma_0 = 1$, we have:

$$U^{(x_0,\gamma_0)} = \begin{bmatrix} \sqrt{0.7} & \sqrt{0.1} - i\sqrt{0.2} \\ -\sqrt{0.1} - i\sqrt{0.2} & \sqrt{0.7} \end{bmatrix}.$$

Thus, as

$$|\psi_{0_2}\rangle = U^{(x_0,\gamma_0)}|\psi_{0_1}\rangle$$

follows

$$|\psi_{0_2}\rangle = (\sqrt{0.1} - i\sqrt{0.2})|0;00\rangle + \sqrt{0.7}|1;00\rangle. \tag{17}$$

Note that in the Eq. (17), the state $|\psi_{0_2}\rangle$ has one more term than $|\psi_{0_1}\rangle$. In the first term, $|u\rangle = |0\rangle$, indicating that the x_0 amplitude has been loaded in this term. While in the second term, $|u\rangle = |1\rangle$, indicating that this term will be used to prepare the next input pattern (x_1, p_1) into quantum state.

The last step in the iteration of the input pattern (x_0, p_0) is not performed either since $t = 0$. Thus, $|\psi_{0_3}\rangle = |\psi_{0_2}\rangle$, and x_0 is loaded as amplitude of the binary pattern p_0 in a quantum state.

For $k = 1$, we will perform the second iteration of the algorithm to prepare the (x_1, p_1) input pattern. As $|\psi_{1_0}\rangle = |\psi_{3_0}\rangle$ we have:

$$|\psi_{1_0}\rangle = (\sqrt{0.1} - i\sqrt{0.2})|0;00\rangle + \sqrt{0.7}|1;00\rangle. \tag{18}$$

For $p_1 = 10$, follow $t = 1$ and $l = \{0\}$. This establishes that in Step 6, we have a single controlled operation given by a $CX_{(u,m[0])}$, controlled by $|u\rangle$ and target in $|m[0]\rangle$, as described below.

$$|\psi_{1_1}\rangle = CX_{(u,m[0])}|\psi_{1_0}\rangle$$

where we obtain

$$|\psi_{1_1}\rangle = (\sqrt{0.1} - i\sqrt{0.2})|0;00\rangle + \sqrt{0.7}|1;10\rangle. \tag{19}$$

Based on t and l, we establish the controlled operation $CU^{(x_1,\gamma_1)}$, controlled by $|m[0]\rangle$ and target in $|u\rangle$. Thus, Step 7 provides:

$$|\psi_{1_2}\rangle = CU^{(x_1,\gamma_1)}{}_{(m[0],u)}|\psi_{1_1}\rangle.$$

Now, we have $\gamma_1 = 1 - 0.3 = 0.7$. By applying $x_1 = \sqrt{0.1}$ and γ_1 in Eq. (5), we obtain:

$$U^{(x_1,\gamma_1)} = \begin{bmatrix} \frac{\sqrt{0.6}}{\sqrt{0.7}} & \frac{\sqrt{0.1}}{\sqrt{0.7}} \\ \\ \frac{-\sqrt{0.1}}{\sqrt{0.7}} & \frac{\sqrt{0.6}}{\sqrt{0.7}} \end{bmatrix}.$$

Obtaining the following quantum state

$$|\psi_{1_2}\rangle = (\sqrt{0.1} - i\sqrt{0.2})|0;00\rangle + \sqrt{0.1}|0;10\rangle + \sqrt{0.6}|1;10\rangle. \tag{20}$$

In the Eq. (20), the x_1 amplitude was loaded into the second term, whose auxiliary qubit were changed to $|u\rangle = |0\rangle$. As there are still patterns to be processed, the operation creates a new term, where $|u\rangle = |1\rangle$, which will be used to process the (x_2, p_2) input pattern.

The last step of this iteration provides:

$$|\psi_{1_3}\rangle = CX_{(u,m[0])}|\psi_{1_2}\rangle$$

reversing the effect of the first controlled operation carried out at the beginning of this iteration, providing:

$$|\psi_{1_3}\rangle = (\sqrt{0.1} - i\sqrt{0.2})|0;00\rangle + \sqrt{0.1}|0;10\rangle + \sqrt{0.6}|1;00\rangle. \quad (21)$$

Now, in the second term of the $|\psi_{1_3}\rangle$ state, $|u\rangle = |0\rangle$, indicating that x_1 is already loaded as amplitude of the p_1 pattern. In contrast, we have $|u\rangle = 1$ in the third term, indicating that the term is in processing and ready to prepare the input pattern (x_2, p_2) into a quantum state.

For the input (x_2, p_2), perform $|\psi_{1_3}\rangle = |\psi_{2_0}\rangle$ followed by the procedure to load x_2 in the amplitude of the patterns p_2 from the state

$$|\psi_{2_0}\rangle = (\sqrt{0.1} - i\sqrt{0.2})|0;00\rangle + \sqrt{0.1}|0;10\rangle + \sqrt{0.6}|1;00\rangle. \quad (22)$$

For $p_2 = 01$, follow $t = 1$ and $l = \{1\}$. This establishes that in Step 6, we have a single controlled operation given by a $CX_{(u,m[1])}$, with control in $|u\rangle$ and target in $|m[1]\rangle$, as described below

$$|\psi_{2_1}\rangle = CX_{(u,m[1])}|\psi_{2_0}\rangle$$

where we obtain

$$|\psi_{2_1}\rangle = (\sqrt{0.1} - i\sqrt{0.2})|0;00\rangle + \sqrt{0.1}|0;10\rangle + \sqrt{0.6}|1;01\rangle. \quad (23)$$

Based on t and l, we establish the controlled operation $CU^{(x_2,\gamma_2)}$, with control in $|m[1]\rangle$ and target in $|u\rangle$. Thus, Step 7 provides:

$$|\psi_{2_2}\rangle = CU^{(x_2,\gamma_2)}{}_{(m[1],u)}|\psi_{2_1}\rangle.$$

Applying $x_2 = \sqrt{0.1} - i\sqrt{0.1}$ and $\gamma_2 = 0.6.$ in Eq. (5), we will get:

$$U^{(x_2,\gamma_2)} = \begin{bmatrix} \frac{\sqrt{0.4}}{\sqrt{0.6}} & \frac{\sqrt{0.1}-i\sqrt{0.1}}{\sqrt{0.6}} \\ \\ \frac{-\sqrt{0.1}+i\sqrt{0.1}}{\sqrt{0.6}} & \frac{\sqrt{0.4}}{\sqrt{0.6}} \end{bmatrix}.$$

Then, by applying the operator $CU^{(x_2,\gamma_2)}$ in the $|\psi_{2_1}\rangle$ state with control in $|m[1]\rangle$ and target in $|u\rangle$, we reach:

$$\begin{aligned} |\psi_{2_2}\rangle &= (\sqrt{0.1} - i\sqrt{0.2})|0;00\rangle + \sqrt{0.1}|0;10\rangle \\ &+ (\sqrt{0.1} - i\sqrt{0.1})|0;01\rangle + \sqrt{0.4})|1;01\rangle. \end{aligned} \quad (24)$$

Finally, the last step of this iteration provides:

$$|\psi_{2_3}\rangle = CX_{(u,m[1])}|\psi_{2_2}\rangle$$

By applying the $CX_{(u,m[1])}$ operator in $|\psi_{2_2}\rangle$ that reverses the first one, we obtain:

$$\begin{aligned} |\psi_{2_3}\rangle &= (\sqrt{0.1} - i\sqrt{0.2})|0;00\rangle + \sqrt{0.1}|0;10\rangle \\ &+ (\sqrt{0.1} - i\sqrt{0.1})|0;01\rangle + \sqrt{0.4})|1;00\rangle. \end{aligned} \quad (25)$$

The third term of the $|\psi_{2_3}\rangle$ state has $|u\rangle = |0\rangle$, indicating that the process of loading x_2 into the p_2 pattern is complete, and the last term will be used to load the state x_3 as the amplitude of the p_3 pattern.

For the last input (x_3, p_3), as $|\psi_{2_3}\rangle = |\psi_{3_0}\rangle$, follow

$$\begin{aligned} |\psi_{3_0}\rangle &= (\sqrt{0.1} - i\sqrt{0.2})|0;00\rangle + \sqrt{0.1}|0;10\rangle \\ &+ (\sqrt{0.1} - i\sqrt{0.1})|0;01\rangle + \sqrt{0.4})|1;00\rangle. \end{aligned} \tag{26}$$

For $p_3 = 11$, according to the Algorithm 2, follow $t = 2$ and $l = \{0, 1\}$. This establishes that in Step 6, we have two sequential controlled operations, one for each $j \in l$, given by $CX_{(u,m[0])}$ and $CX_{(u,m[0])}$, as described below

$$|\psi_{3_1}\rangle = CX_{(u,m[0])}.CX_{(u,m[1])}|\psi_{3_0}\rangle$$

where we obtain

$$\begin{aligned} |\psi_{3_1}\rangle &= (\sqrt{0.1} - i\sqrt{0.2})|0;00\rangle + \sqrt{0.1}|0;10\rangle \\ &+ (\sqrt{0.1} - i\sqrt{0.1})|0;01\rangle + \sqrt{0.4})|1;11\rangle. \end{aligned} \tag{27}$$

Based on t and l, we establish the controlled operation $C^2U^{(x_3,\gamma_3)}$, with controls in $|m[0]\rangle$ and $|m[1]\rangle$, and target in $|u\rangle$. Thus, Step 7 provides:

$$|\psi_{3_2}\rangle = C^2U^{(x_2,\gamma_2)}{}_{(m[0]m[1],u)}|\psi_{3_1}\rangle.$$

By applying $x_3 = \sqrt{0.4}$ and $\gamma_3 = 0.4$ in Eq. (5), we obtain:

$$U^{(x_3,\gamma_3)} = \begin{bmatrix} 0 & 1 \\ \\ -1 & 0 \end{bmatrix}.$$

Thus, we reach the following state

$$\begin{aligned} |\psi_{3_2}\rangle &= (\sqrt{0.1} - i\sqrt{0.2})|0;00\rangle + \sqrt{0.1}|0;10\rangle \\ &+ (\sqrt{0.1} - i\sqrt{0.1})|0;01\rangle + \sqrt{0.4}|0;11\rangle. \end{aligned} \tag{28}$$

Note that as this is the last iteration in the Algorithm 2, after applying the operation $U^{(x_3,\gamma_3)}$, the last term does not create a superposition on $|u\rangle$, only the auxiliary qubit has its status set to $|u\rangle = |0\rangle$. This indicates that the amplitude x_3 was loaded in the last term of the quantum state in preparation. The last step of the last iteration does not need to be performed as all input patterns (x_k, p_k) are loaded, and there are no more terms in the process. Thus, we can rewrite the $|\psi_{3_2}\rangle$ state as follows:

$$|\psi_{3_2}\rangle = |0\rangle \otimes \big(x_0|00\rangle + x_1|10\rangle + x_2|01\rangle + x_3|11\rangle\big). \tag{29}$$

As $|u\rangle = |0\rangle$ in all terms, indicating that the execution of the algorithm is finished. Therefore, $\mathcal{D}$ is prepared as the desired quantum state in the Eq. (16):

$$\begin{aligned}|\psi\rangle =&(\sqrt{0.1} - i\sqrt{0.2})|00\rangle + (\sqrt{0.1} - i\sqrt{0.1})|01\rangle \\ &+\sqrt{0.1}|10\rangle + \sqrt{0.4}|11\rangle. \end{aligned} \tag{30}$$

4 Results and Experiments

We compared the CNOT cost resulting from the CVO-QRAM algorithm for state preparation presented in Section 3 to the dense and sparse scenarios. The state preparation algorithm is described in more detail in Subsection 3.3. All of the algorithms are implemented in *Qiskit* [29] using the combination of [5] and [28] to implement gates of the type C^nU mentioned in the sub-subsection 1.1.1. The references of each implementation are given in Table 2.

4.1 Dense quantum state preparation

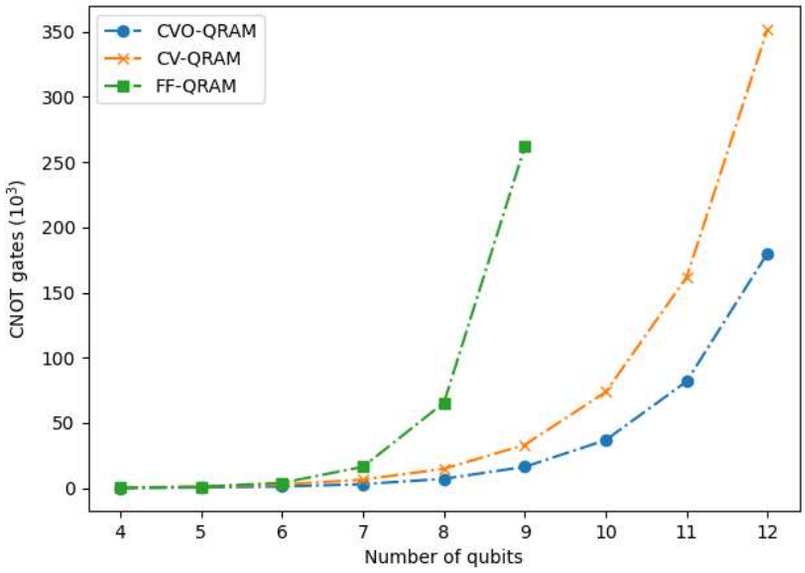
Table 1 summarizes the number of CNOT gates required by the quantum algorithms to prepare a dense quantum state with n qubits and continuous amplitudes from an input data set $\mathcal{D}$ containing $M = 2^n$ inputs of type (x_k, p_k) where $p_k \in \{0, 1\}^n$.

We performed experiments using the CVO-QRAM, CV-QRAM, and FF-QRAM to prepare a dense quantum state. Figure 7 presents the results. The FF-QRAM algorithm uses significantly more CNOTs than the CVO-QRAM. The advantage of the CV-QRAM algorithm is less significant but also indicates that the desired optimization was obtained for the preparation of dense quantum state.

Algorithm	CNOT count	Reference
CVO-QRAM	$\sum_{t=1}^{n} C(n,t)(8t-4) - n$	Section 3
CV-QRAM	$2^n(8n-2)$	T. M. L. Veras *et al.* [19]
UGD	$< \frac{23}{24}2^n$	Martin Plesch *et al.* [25]
SQL	$2^{n+1} - 2n$	Vivek V Shende *et al.* [17]
Isometry	$< \frac{23}{32}2^n$	Raban Iten *et al.* [30]
Möttönen	$2^{n+2} - 4n - 4$	Mikko Möttönen *et al.* [23]
FF-QRAM	$2^n(6n-4)$	Daniel K. Park *et al.* [18]

Table 1: Number of CNOT gates required to prepare a dense quantum state of n qubits with $M = 2^n$ input data.

Figure 7: CNOT count to prepare a dense quantum state of n qubits with $M = 2^4$ input data.



4.2 Sparse quantum state preparation

Table 2 presents the number of CNOTs gates needed to prepare a sparse quantum state with n qubits and $M = 2^4$ non-zero inputs using the aforementioned algorithms. In this experiment, the set of binary patterns is composed of about 50% values equal to 1. According to (3.4.2), such percentage of 1s in the binary string sets the worst scenario for the CVO-QRAM algorithm. Note that as n grows, the prepared quantum state becomes increasingly sparse.

The CVO-QRAM algorithm causes a clear improvement of the CV-QRAM in relation to the CNOT gates required to prepare the desired quantum state. Figure 8 illustrates the results obtained to prepare a sparse quantum state with 2^4 non-zero amplitudes as n grows.

Algorithm	Script	$n = 6$	$n = 7$	$n = 8$	$n = 9$	$n = 10$	$n = 11$	$n = 12$
CVO-QRAM	Ref. [31]	275	466	474	552	**542**	**742**	**719**
CV-QRAM	Ref. [31]	694	838	936	1052	1146	1292	1382
UGD	Ref. [31]	**53**	150	248	657	1089	2768	4475
SQL	Ref. [29]	62	126	254	510	1022	2046	4094
Isometry	Ref. [29]	57	**120**	**247**	**502**	1013	2036	4083
Möttönen	Ref. [32]	124	252	508	1020	2044	4092	8188

Table 2: Performance of sparse state preparation. In each column, we highlight the lowest CNOT count to prepare a sparse quantum state of n qubits with $M = 2^4$ input data.

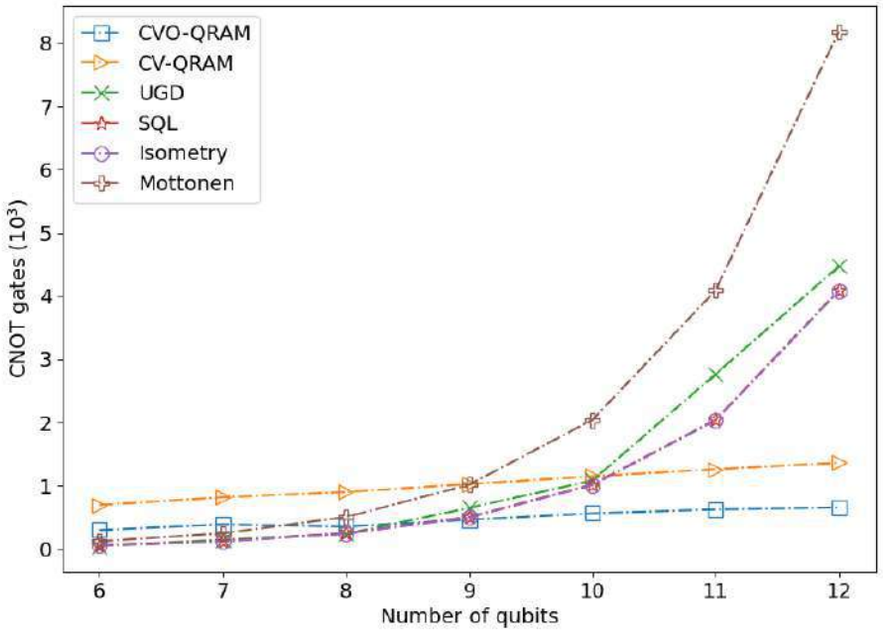


Figure 8: Performance of sparse state preparation. CNOT count to prepare a sparse quantum state of n qubits with $M = 2^4$ input data.

Recently, [27] has proposed an algorithm that deals with the preparation of sparse quantum states. Figure 9 shows simulations comparing the CVO-QRAM and [27] algorithms for the preparation of double sparse quantum states (sparse regarding the number of patterns and number of 1s in the binary strings) with n qubits, containing 2^s non-zero amplitudes where $s < n$. Both algorithms use the multi-controlled quantum gate decomposition described in Sec 1.1 and are implemented in Qiskit [29].

By fixing a number of non-zero amplitudes 2^s, the experiments show that the CVO-QRAM algorithm requires fewer CNOT gates when the number of qubits $n \gg 1$, with even more favorable results when s grows. Furthermore, the classical computational cost in [27] is quadratic regarding the number of non-zero amplitudes compared to the CVO-QRAM, which has a classical cost $\mathcal{O}(M \log M + Mn)$.

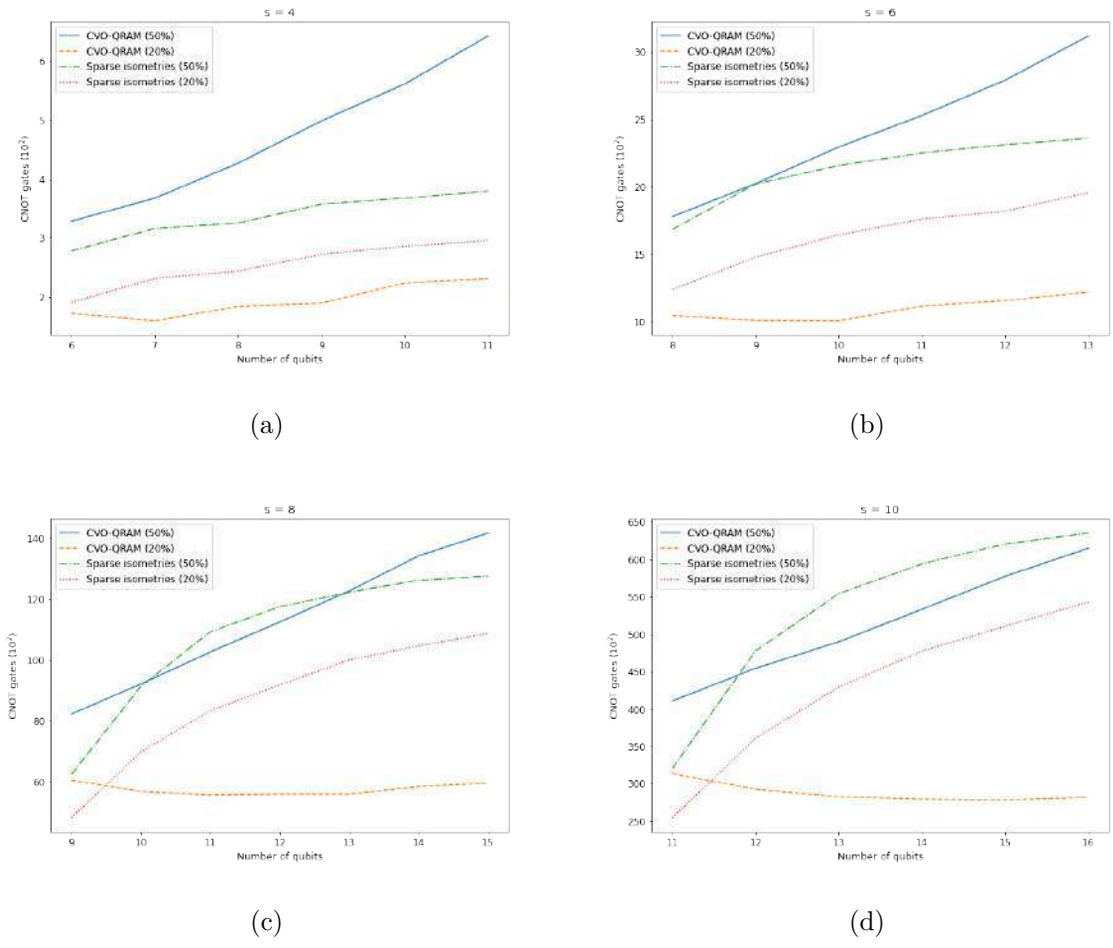


Figure 9: This shows the average number of CNOT gates produced by CVO-QRAM and the Sparse isometry algorithm [27] for sparse states on n qubits with 2^s non-zero entries, whose positions for bit 1 are chosen uniformly and randomly. **Solid and dashdot lines**. CNOT count for a binary pattern set consisting of about 50% values equal to 1. **Dashed and dotted lines**. CNOT count for a binary pattern set consisting of at most 20% values equal to 1. We used 100 randomly generated samples to average.

5 Conclusion

The ability to interpret classical data and load it into a quantum state efficiently is a critical process in quantum computing that cannot be overlooked.

The work [19] proposes an algorithm for preparing quantum states called CV-QRAM that loads a database with even elements, composed of a complex number and a binary pattern with n bits. The cost of the [19] algorithm depends on the number of input patterns and qubits, requiring $8n - 4$ CNOT gates to load an input pattern in the quantum state, thus representing a promising algorithm for preparing sparse quantum states.

This paper introduces a quantum state preparation algorithm, designated as CVO-QRAM, to optimize the CV-QRAM. The cost in CVO-QRAM is $8t - 4$ CNOT gates to load an input pattern into the quantum state, where t is the amount of 1s in the binary string of the pattern being stored. Thus, CVO-QRAM depends on the numbers of input patterns, qubits, and 1s in

the binary string.

The CVO-QRAM proved more efficient than [18], [19] in all scenarios; in addition, as the experiments demonstrated, it is quite competitive in the preparation of sparse quantum states compared to other state preparation algorithms [17], [23], [25], [30] when $n \gg 1$. If the desired quantum state is double sparse, then the CVO-QRAM provides better results than [27] for a fixed number of non-zero amplitudes s and $n \gg 1$. The results are even more favorable when s grows and $n \gg 1$. Therefore, CVO-QRAM is an efficient quantum state preparation algorithm in preparing sparse quantum states.

References

[1] Richard P. Feynman. Simulating physics with computers. Int. J. Theor. Phys., 21:467–488, 1982.

[2] D. Deutsch. Quantum theory, the church–turing principle and the universal quantum computer. Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences, 400:117 – 97, 1985.

[3] David Deutsch and Richard Jozsa. Rapid Solution of Problems by Quantum Computation. Proceedings of the Royal Society of London Series A, 439(1907):553–558, December 1992.

[4] Andrew Steane. Quantum computing. Reports on Progress in Physics, 61(2):117–173, Feb 1998.

[5] Michael A Nielsen and Isaac Chuang. Quantum computation and quantum information. AAPT, 2002.

[6] Noson S. Yanofsky and Mirco A. Mannucci. Quantum Computing for Computer Scientists. Cambridge University Press, USA, 1 edition, 2008.

[7] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM review, 41(2):303–332, 1999.

[8] Lov K Grover. Quantum mechanics helps in searching for a needle in a haystack. Physical review letters, 79(2):325, 1997.

[9] Dan Ventura and Tony Martinez. Initializing the amplitude distribution of a quantum state. Foundations of Physics Letters, 12(6):547–559, 1999.

[10] Gui-Lu Long and Yang Sun. Efficient scheme for initializing a quantum register with an arbitrary superposed state. Physical Review A, 64(1), Jun 2001.

[11] Lov K Grover. Synthesis of quantum superpositions by quantum computation. Physical review letters, 85(6):1334, 2000.

[12] Maria Schuld and Francesco Petruccione. Supervised learning with quantum computers, volume 17. Springer, 2018.

[13] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. Quantum machine learning. Nature, 549, 11 2016.

[14] Aram W Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. Physical review letters, 103(15):150502, 2009.

[15] Dominic W Berry and Andrew M Childs. Black-box hamiltonian simulation and unitary implementation. Quantum Information & Computation, 12(1-2):29–62, 2012.

[16] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. Quantum random access memory. Physical Review Letters, 100(16), Apr 2008.

[17] V.V. Shende, S.S. Bullock, and I.L. Markov. Synthesis of quantum-logic circuits. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 25(6):1000–1010, Jun 2006.

[18] Daniel K. Park, Francesco Petruccione, and June-Koo Kevin Rhee. Circuit-based quantum random access memory for classical data. Scientific Reports, 9(1), 2019.

[19] T. M. L. Veras, I. C. S. De Araujo, K. D. Park, and A. J. da Silva. Circuit-based quantum random access memory for classical data with continuous amplitudes. IEEE Transactions on Computers, pages 1–1, 2020.

[20] Andrei N. Soklakov and Rüdiger Schack. Efficient state preparation for a register of quantum bits. Physical Review A, 73(1), jan 2006.

[21] Yuval R. Sanders, Guang Hao Low, Artur Scherer, and Dominic W. Berry. Black-box quantum state preparation without arithmetic. Physical Review Letters, 122(2), jan 2019.

[22] Johannes Bausch. Fast black-box quantum state preparation, 2020.

[23] Mikko Möttönen, Juha J Vartiainen, Ville Bergholm, and Martti M Salomaa. Transformation of quantum states using uniformly controlled rotations. Quantum Information & Computation, 5(6):467–473, 2005.

[24] Vivek V Shende, Stephen S Bullock, and Igor L Markov. Synthesis of quantum-logic circuits. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 25(6):1000–1010, 2006.

[25] Martin Plesch and Časlav Brukner. Quantum-state preparation with universal gate decompositions. Physical Review A, 83(3):032302, 2011.

[26] Carlo A Trugenberger. Probabilistic quantum memories. Physical Review Letters, 87(6):067901, 2001.

[27] Emanuel Malvetti, Raban Iten, and Roger Colbeck. Quantum circuits for sparse isometries. Quantum, 5:412, 2021.

[28] Adriano Barenco, Charles H Bennett, Richard Cleve, David P DiVincenzo, Norman Margolus, Peter Shor, Tycho Sleator, John A Smolin, and Harald Weinfurter. Elementary gates for quantum computation. Physical review A, 52(5):3457, 1995.

[29] Gadi Aleksandrowicz and et al. Qiskit: An open-source framework for quantum computing, 2019.

[30] Raban Iten, Roger Colbeck, Ivan Kukuljan, Jonathan Home, and Matthias Christandl. Quantum circuits for isometries. Physical Review A, 93(3), mar 2016.

[31] Adenilton J. da Silva and et al. Quantum computing library. `https://github.com/qclib/qclib`, 2021.

[32] Carsten Blank. Data cybernetics qiskit-algorithms. `https://github.com/carstenblank/dc-qiskit-algorithms`, 2021.

5 CONCLUSION

Receiving a classical dataset as input and providing a quantum state representing this classical dataset as output is a critical point in current quantum computing. Many quantum algorithms require, as initialization, a quantum state prepared from a set of classical data. Such process of converting classical data into a quantum state is called quantum state preparation.

Many quantum algorithms requires several measurements during execution, leading the initial state to be recreated several times. Thus, an efficient performance of quantum states preparation is required to prevent the efficiency of these quantum algorithms to comprised. This scenario makes quantum state preparation an essential subroutine in building an efficient quantum algorithm. In this sense, the results of this dissertation introduce three algorithms for sparse quantum state preparation. The first two, called Preprocessed FFP-QRAM and CV-QRAM, solve our first two specific goals, whereas the third is called CVO-RAM and solves our third specific goal and the main objective of this research.

The Preprocessed FFP-QRAM algorithm was built to improve the post-selection of the FF-QRAM and load continuous amplitudes in the quantum state. By preprocessing on the input amplitudes and combining the PQM with FF-QRAM, our algorithm resulted in a probabilistic algorithm that loads continuous amplitude with better post-selection probability than FF-QRAM.

At the beginning of this research, there were no specific algorithms for preparing sparse quantum states. The main algorithms for preparing states known at computational cost depend on the number of qubits. This cost is exponential in the number of steps and exponential in the classical computer. The issue that remains open is how to obtain an algorithm for preparing sparse quantum states without requiring post-selection at computational cost $\mathcal{O}(Mn)$ steps and classical cost $\mathcal{O}(n \log n)$, where M is the number of input patterns and n the number of qubits.

The FF-QRAM is a probabilistic algorithm with a cost of $\mathcal{O}(CMn)$ steps and has a classical cost of $\mathcal{O}(n)$ to have the circuit built, where C is the number of repetitions for post-selection that depends on the data distribution. In the first paper, entitled "Circuit-based quantum random access memory for classical data with continuous amplitudes", we showed that FF-QRAM is inefficient.

The CV-QRAM was built particularly to eliminate the post-selection existing in the FF-QRAM (PARK; PETRUCCIONE; RHEE, 2019) algorithm, load continuous amplitudes, at a computational cost of $\mathcal{O}(Mn)$ steps and classical cost $\mathcal{O}(Mn)$ to have the circuit built. Such goal was achieved, and CV-QRAM achieved more efficient results than the FF-QRAM.

In contrast, even removing the post-selection, CV-QRAM algorithm still has a cost of $8n-2$ CNOT gates to load each input pattern in the quantum state, which depends on the number of input patterns and the number of qubits. Reducing the number of CNOT gates used can contribute to the efficiency of a quantum algorithm since the implementation of this quantum gate produces more noise in the results.

In order to build a quantum state preparation algorithm capable of loading continuous amplitudes without post-selection and reducing the number of CNOT gates needed in relation to CV-QRAM, we obtained a quantum state preparation algorithm called CVO-QRAM that optimizes the CV-QRAM and solves our third specific research problem. The CVO-QRAM requires $8t - 4$ CNOT gates to load each input pattern into the quantum state, where t is the amount of 1s in the binary string of the pattern being stored. Thus CVO-QRAM depends on the number of input patterns, qubits, and 1s in the binary string.

The CVO-QRAM proved to be more efficient than (PARK; PETRUCCIONE; RHEE, 2019), (VERAS et al., 2020) in all scenarios, in addition to proving, as shown in the experiments, to be quite competitive in the preparation of sparse quantum states in relation to other state preparation algorithms (SHENDE; BULLOCK; MARKOV, 2006), (MÖTTÖNEN et al., 2005), (PLESCH; BRUKNER, 2011), (ITEN et al., 2016) when $n \gg 1$.

Recently, a sparse quantum state preparation algorithm was proposed by Malvetti, Iten e Colbeck (2021). The CVO-QRAM is deterministic when loading continuous amplitudes at computational cost of $O(Mn)$ steps and classical cost $\mathcal{O}(M \log M + Mn)$ to build the circuit regarding the number of non-zero amplitudes, while the pivoting algorithm (MALVETTI; ITEN; COLBECK, 2021) has a classical cost $\mathcal{O}(M^2)$.

The experiments demonstrate that to prepare a double sparse quantum state (sparse regarding the number of patterns and the binary string) with n qubits, containing $M = 2^s$ non-zero amplitudes, where $s < n$. If the quantum state desired is double sparse, then CVO-QRAM provides better results than (MALVETTI; ITEN; COLBECK, 2021) for a fixed number of non-zero amplitudes s and $n \gg 1$. The results are even more favorable when s grows and $n \gg 1$. Therefore, CVO-QRAM is an efficient quantum state preparation algorithm in preparing sparse quantum states.

5.1 MAIN CONTRIBUTIONS

In the study entitled **Circuit-based quantum random access memory for classical data with continuous amplitudes** and published in **IEEE Transactions on Computers**, we achieved the results that solve our first goal, also providing our two first contributions to quantum states preparation.

- We improved the post-selection probability of FF-QRAM through the pre-processed FFP-QRAM algorithm.

- We obtained a quantum state preparation algorithm, called CV-QRAM, capable of loading continuous amplitudes into a quantum state, eliminating the post-selection required in FF-QRAM.

In the preprint (VERAS; SILVA; DA SILVA, 2021) entitled **Double sparse quantum state preparation**, we built the CVO-QRAM algorithm that solves our third specific goal and the main research problem, as described below:

- The CVO-QRAM algorithm is a deterministic algorithm that loads continuous amplitudes at computational cost of $O(Mn)$ steps and circuit construction cost of $\mathcal{O}(M \log M + Mn)$ regarding the number of non-zero amplitudes. The CVO-QRAM presents better results in sparse quantum states preparation with better efficiency upon a large number of non-zero amplitudes.

  Regarding the cost in the number of CNOT gates required, the CVO-QRAM presented better results in the preparation of sparse quantum states than the methods proposed in the literature when $n \gg 1$. If the desired state is double sparse, CVO-QRAM is more efficient than (MALVETTI; ITEN; COLBECK, 2021) for a fixed number of non-zero amplitudes s and $n \gg 1$, with even more favorable results as s grows.

5.2 FURTHER RESEARCH

Since the storage order established in CVO-QRAM does not consider a better order, it is possible to eliminate some CNOT gates, thus representing some disadvantages of the CVO-QRAM regarding some (PLESCH; BRUKNER, 2011), (MÖTTÖNEN et al., 2005), (SHENDE; BULLOCK; MARKOV, 2006) algorithms in the preparation of dense quantum state. In order to improve the performance of CVO-QRAM for dense cases, we present the following further work.

- **Further research 1**

The number of CNOT gates needed for the CVO-QRAM algorithm to prepare a quantum state can be improved if we can establish what we define as a better storage order. This is equivalent to establishing a storage order in which any two patterns p_k and p_{k+1} had the greatest amount of 1s in the binary string in the same positions. This would result in the largest possible cancellation of CNOT gates among the patterns and hence the entire algorithm.

- **Further research 2**

The pivoting algorithm proposed in (MALVETTI; ITEN; COLBECK, 2021) prepares a sparse quantum state containing $M = 2^s$ non-zero amplitudes in a quantum state, with n

qubits where $s < n$, $M-1$ controlled gates of type C^sX are required to load the patterns in a quantum state. For $n \gg 1$, the CNOT cost of the algorithm in the preparation of sparse states is directly associated with the value of s due to the s-controlled gates. From this observation, if we combine the CVO-QRAM algorithm with the pivoting algorithm to prepare sparse quantum states, we hope to improve the cost of CNOT gates needed to prepare a sparse quantum state. For this purpose, we will prepare the initial quantum state of the proposed algorithm in (MALVETTI; ITEN; COLBECK, 2021) using the best case in CVO-QRAM. This provides us with an initial state prepared with the maximum number of 0s in the binary string of the patterns, requiring the minimum number of CNOT gates enabled by the CVO-QRAM. Next, we use pivoting from the initial state to the desired state expecting a lower cost of CNOT gates.

- **Further research 3**

The number of CNOT gates required to load an input pattern (x_k, p_k) into a quantum state using CVO-QRAM is determined by the number of 1s in the binary string. Thus, if we build an algorithm that takes a binary pattern p_k full of 1s and resets this pattern before loading, with the help of ancillas and X or CNOT operations in the state, storing is required. We can store that binary pattern as if it were a pattern in which all binary strings were zero. This would disregard multi-control ports with a large number of controls, which, in turn, would always be performed by ancillas. Thus, even for a dense quantum state and number of qubits $n \gg 1$, we would not required an excess of operators of the type C^tX, where t is the number of 1s in the binary string, which would reduce considerably the number of CNOT gates needed. From the quantum algorithm to solve linear systems of equations proposed by (HARROW; HASSIDIM; LLOYD, 2009), quantum machine learning (QML) has been shown to be a viable way to produce an exponential acceleration in the field of machine learning.

Algorithms like (HARROW; HASSIDIM; LLOYD, 2009), (KERENIDIS; PRAKASH, 2016), (LLOYD; MOHSENI; REBENTROST, 2013), (LI et al., 2021) require a quantum state preparation, the argument used from an arbitrary input vector can prepare the desired quantum state efficiently. Thus, these algorithms may achieve an exponential acceleration for assuming that the state is efficiently prepared without accounting for the preparation cost in the main cost of the (TANG, 2021) algorithm. These algorithms neglect the cost of state by assuming an efficient preparation; in addition, as demonstrated in this dissertation, this preparation can compromise the efficiency of an algorithm. Therefore, we will verify if algorithms that claim to be efficient are indeed still efficient, assuming that their initial state has been prepared efficiently.

REFERENCES

ARAÚJO, I.; PARK, D.; PETRUCCIONE, F.; DA SILVA, A. J. A divide-and-conquer algorithm for quantum state preparation. *Scientific Reports*, v. 11, 2021.

ARUNACHALAM, S.; GHEORGHIU, V.; JOCHYM-O'CONNOR, T.; MOSCA, M.; SRINIVASAN, P. On the robustness of bucket brigade quantum RAM. *New Journal of Physics*, v. 17, p. 123010, 2015.

BARENCO, A.; BENNETT, C.; CLEVE, R.; DIVINCENZO, D.; MARGOLUS, N.; SHOR, P.; SLEATOR, T.; SMOLIN, J.; WEINFURTER, H. Elementary gates for quantum computation. *Physical Review A*, v. 52, p. 3457, 1995.

BASSI, A.; DECKERT, D.-A. Noise gates for decoherent quantum circuits. *Physical Review A*, American Physical Society (APS), v. 77, n. 3, p. 032323, 2008.

BAUSCH, J. Fast black-box quantum state preparation. *arXiv preprint arXiv:2009.10709*, 2020.

BENIOFF, P. The computer as a physical system: A microscopic quantum mechanical hamiltonian model of computers as represented by turing machines. *Journal of Statistical Physics*, v. 22, p. 563–591, 1980.

BERRY, D.; CHILDS, A.; CLEVE, R.; KOTHARI, R.; SOMMA, R. Simulating hamiltonian dynamics with a truncated taylor series. *Physical review letters*, v. 114, p. 090502, 2014.

BERRY, D.; CHILDS, A.; OSTRANDER, A.; WANG, G. Quantum algorithm for linear differential equations with exponentially improved dependence on precision. *Communications in Mathematical Physics*, v. 356, p. 1057–1081, 2017.

BIAMONTE, J.; WITTEK, P.; PANCOTTI, N.; REBENTROST, P.; WIEBE, N.; LLOYD, S. Quantum machine learning. *Nature*, v. 549, p. 195–202, 2016.

CHILDS, A. On the relationship between continuous- and discrete-time quantum walk. *Commun. Math. Phys.*, v. 294, p. 581–603, 2008.

CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C. *Introduction to Algorithms, Third Edition*. 3rd. ed. Cambridge: The MIT Press, 2009. ISBN 0262033844.

DEUTSCH, D. Quantum theory, the church-turing principle and the universal quantum computer. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, v. 400, p. 97–117, 1985.

DEUTSCH, D.; JOZSA, R. Rapid solution of problems by quantum computation. *Proc. Roy. Soc. Lond. A*, v. 439, p. 553–558, 12 1992.

FEYNMAN, R. *Feynman Lectures on Computation*. New York: Perseus Books Group, 2000.

FEYNMAN, R. P. Simulating physics with computers. *Int. J. Theor. Phys.*, v. 21, p. 467–488, 1982.

GIOVANNETTI, V.; LLOYD, S.; MACCONE, L. Quantum random access memory. *Physical review letters*, v. 100, p. 160501, 2008.

GIROLAMI, D. How difficult is it to prepare a quantum state? *Physical Review Letters*, American Physical Society (APS), v. 122, n. 1, 2019.

GROVER, L. Quantum mechanics helps in searching for a needle in a haystack. *Physical Review Letters*, v. 79, 1997.

GROVER, L. Quantum computers can search rapidly by using almost any transformation. *Physical Review Letters*, v. 80, 1998.

GROVER, L. Synthesis of quantum superpositions by quantum computation. *Physical review letters*, v. 85, p. 1334–7, 2000.

HARROW, A.; HASSIDIM, A.; LLOYD, S. Quantum algorithm for linear systems of equations. *Physical review letters*, v. 103, p. 150502, 2009.

HUGHES, R.; JAMES, D.; KNILL, E.; LAFLAMME, R.; PETSCHEK, A. Decoherence bounds on quantum computation with trapped ions. *Physical review letters*, v. 77, p. 3240–3243, 1996.

ITEN, R.; COLBECK, R.; KUKULJAN, I.; HOME, J.; CHRISTANDL, M. Quantum circuits for isometries. *Physical Review A*, American Physical Society (APS), v. 93, n. 3, 2016.

JAEGERS, R.; BLALOCK, T. *Microelectronic circuit design*. New York: McGraw-Hill education, 2003.

KAYE, P.; MOSCA, M. Quantum networks for generating arbitrary quantum states. In: OPTICAL SOCIETY OF AMERICA. *International Conference on Quantum Information*. New York, 2001. p. PB28.

KERENIDIS, I.; PRAKASH, A. Quantum recommendation systems. *arXiv preprint arXiv:1603.08675*, 2016.

KIEFEROVA, M.; WIEBE, N. Tomography and generative training with quantum boltzmann machines. *Physical Review A*, v. 96, 12 2017.

KOTHARI, R. Efficient algorithms in quantum query complexity. UWSpace, 2014.

LAROSE, R.; COYLE, B. Robust data encodings for quantum classifiers. *Physical Review A*, American Physical Society (APS), v. 102, n. 3, 2020.

LI, Z.; CHAI, Z.; GUO, Y.; JI, W.; WANG, M.; SHI, F.; WANG, Y.; LLOYD, S.; DU, J. Resonant quantum principal component analysis. *Science Advances*, v. 7, p. eabg2589, 2021.

LLOYD, S. Universal quantum simulators. *Science (New York, N.Y.)*, v. 273, p. 1073–8, 1996.

LLOYD, S.; MOHSENI, M.; REBENTROST, P. Quantum algorithms for supervised and unsupervised machine learning. 2013.

LONG, G.; SUN, Y. Efficient scheme for initializing a quantum register with an arbitrary superposed state. *Physical Review A*, v. 64, 2001.

LOW, G. H.; CHUANG, I. Optimal hamiltonian simulation by quantum signal processing. *Physical Review Letters*, v. 118, p. 010501, 2016.

MALVETTI, E.; ITEN, R.; COLBECK, R. Quantum circuits for sparse isometries. *Quantum*, v. 5, p. 412, 2021.

MATTEO, O.; GHEORGHIU, V.; MOSCA, M. Fault-tolerant resource estimation of quantum random-access memories. *IEEE Transactions on Quantum Engineering*, v. 1, p. 1–13, 2020.

MÖTTÖNEN, M.; VARTIAINEN, J. J.; BERGHOLM, V.; SALOMAA, M. M. Transformation of quantum states using uniformly controlled rotations. *Quantum Information & Computation*, Rinton Press, Incorporated, v. 5, n. 6, p. 467–473, 2005.

NAKAHARA, M. *Quantum computing: from linear algebra to physical realizations*. New York: CRC press, 2008.

NIELSEN, M. A.; CHUANG, I. *Quantum computation and quantum information*. [S.l.]: AAPT, 2002.

PALER, A.; OUMAROU, O.; BASMADJIAN, R. Parallelizing the queries in a bucket-brigade quantum random access memory. *Physical Review A*, v. 102, 2020.

PARK, D. K.; PETRUCCIONE, F.; RHEE, J.-K. K. Circuit-based quantum random access memory for classical data. *Scientific Reports*, Springer Science and Business Media LLC, v. 9, n. 1, 2019.

PLESCH, M.; BRUKNER, Č. Quantum-state preparation with universal gate decompositions. *Physical Review A*, American Physical Society (APS), v. 83, n. 3, 2011.

PRESKILL, J. Quantum computing in the nisq era and beyond. *Quantum*, v. 2, 2018.

RADIOHEAD. *Weird Fishes / Arpeggi*. 2003.

SANDERS, Y. R.; LOW, G. H.; SCHERER, A.; BERRY, D. W. Black-box quantum state preparation without arithmetic. *Physical Review Letters*, American Physical Society (APS), v. 122, n. 2, jan 2019.

SCHERER, A.; VALIRON, B.; MAU, S.-C.; ALEXANDER, S.; BERG, E.; CHAPURAN, T. Concrete resource analysis of the quantum linear system algorithm used to compute the electromagnetic scattering cross section of a 2d target. *Quantum Information Processing*, v. 16, 2017.

SCHULD, M.; KILLORAN, N. Quantum machine learning in feature hilbert spaces. *Physical Review Letters*, v. 122, 2018.

SHENDE, V.; BULLOCK, S.; MARKOV, I. Synthesis of quantum logic circuits. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, v. 25, p. 1000 – 1010, 2006.

SHOR, P. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Review*, v. 41, 1997.

SOKLAKOV, A.; SCHACK, R. State preparation based on grover's algorithm in the presence of global information about the state. *Optics and Spectroscopy*, v. 99, p. 211–217, 2005.

SOMMA, R.; BOIXO, S.; BARNUM, H.; KNILL, E. Quantum simulations of classical annealing processes. *Physical review letters*, v. 101, p. 130504, 2008.

TANG, E. Quantum principal component analysis only achieves an exponential speedup because of its state preparation assumptions. *Physical Review Letters*, v. 127, 2021.

TRUGENBERGER, C. A. Probabilistic quantum memories. *Physical Review Letters*, American Physical Society (APS), v. 87, n. 6, p. 067901, 2001.

VARTIAINEN, J.; MÖTTÖNEN, M.; SALOMAA, M. Efficient decomposition of quantum gates. *Physical review letters*, v. 92, p. 177902, 2004.

VENTURA, D.; MARTINEZ, T. Initializing the amplitude distribution of a quantum state. *Foundations of Physics Letters*, v. 12, p. 547–559, 1970.

VERAS, T. M. L.; ARAUJO, I. C. S. de; PARK, K. D.; DA SILVA, A. J. Circuit-based quantum random access memory for classical data with continuous amplitudes. *IEEE Transactions on Computers*, p. 1–1, 2020.

VERAS, T. M. L. de; SILVA, L. D. da; DA SILVA, A. J. Double sparse quantum state preparation. *arXiv preprint arXiv:2108.13527*, 2021.

WOOTTERS, W.; ZUREK, W. A single quantum cannot be cloned. *Nature*, v. 299, p. 802, 1982.

YANOFSKY, N. S.; MANNUCCI, M. A.; MANNUCCI, M. A. *Quantum computing for computer scientists*. New York: Cambridge University Press Cambridge, 2008. v. 20.

ZALKA, C. Simulating quantum systems on a quantum computer. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, v. 454, n. 1969, p. 313–322, 1998.

APPENDIX A – ARTICLE PUBLISHED IN NEUROCOMPUTING.

Parametric Probabilistic Quantum Memory

Rodrigo S. Sousa[b], Priscila G. M. dos Santos[b], Tiago M.L. Veras[a], Wilson R. de Oliveira[c], Adenilton J. da Silva[a]

[a] *Centro de Informática, Universidade Federal de Pernambuco*
[b] *Departamento de Computação, Universidade Federal Rural de Pernambuco*
[c] *Departamento de Estatística e Informática, Universidade Federal Rural de Pernambuco*

Abstract

Probabilistic Quantum Memory (PQM) is a data structure that computes the distance from a binary input to all binary patterns stored in superposition on the memory. This data structure allows the development of heuristics to speed up artificial neural networks architecture selection. In this work, we propose an improved parametric version of the PQM to perform pattern classification, and we also present a PQM quantum circuit suitable for Noisy Intermediate Scale Quantum (NISQ) computers. We present a classical evaluation of a parametric PQM network classifier on public benchmark datasets. We also perform experiments to verify the viability of PQM on a 5-qubit quantum computer.

Keywords: Quantum computing, Probabilistic quantum memory, Machine learning

1. Introduction

Quantum Computing is a computational paradigm that has been harvesting increasing attention for decades now. Several quantum algorithms have time advantages over their best known classical counterparts [1, 2, 3, 4]. The current advances in quantum hardware are bringing us to the era of Noisy Intermediate-Scale Quantum (NISQ) computers [5]. The quest for quantum supremacy is the search for an efficient solution of a task in a quantum computer that current classical computers are not able to efficiently solve. Some authors argue that given the current state of the art, we will achieve quantum supremacy in the next few years [6]. One of the approaches to achieve this supremacy and to expand the potential applications of quantum computers is through quantum machine learning [7].

Machine learning (ML) [8] aims at developing automated ways for computers to learn a specific task from a given set of data samples. ML has several applications in image classification [9], NP-hard problems [10] and others. Some of the limitations of ML are the lack of algorithms to select an ML method and to avoid local minimums.

Quantum machine learning investigates the use of quantum computing concepts to build quantum-enhanced machine learning models able to outperform the classical ones [7]. For instance, quantum computing can be used to efficiently convert a nonlinearly separable dataset into a linearly separable one by exploiting the exponential size of quantum spaces [11] and to select neural networks architectures without dependence of weights initialization [12, 13]. Quantum machine learning proposes speedups of some algorithms used in machine learning such as reinforcement learning [14] and systems of linear equations [15].

There are several works proposing quantum machine learning models such as a quantum generalization of a neural network [16], a quantum distance-based classifier [17], and quantum associative memories [18, 19, 20, 21]. In [20, 21] is introduced a Probabilistic Quantum Memory (PQM) that computes the Hamming

*A. J. da Silva
Email address: ajsilva@cin.ufpe.br (Adenilton J. da Silva)

distance from an n-bits input pattern to k n-bits patterns with computational cost linear in the number of bits, $O(n)$.

However, the PQM [20] (and quantum associative memories in general) has received criticism from various authors. The memory state is lost in every execution of the retrieval algorithm when the memory state is measured. Memory collapse is considered to be the main limitation of this memory [22] as argued in [23] since the $O(m)$ cost to reload m patterns in the memory using its storage algorithm jeopardizes the PQM advantages. Another author [24] claims that the probabilistic quantum memory cannot be considered as a complete model. However, in [4] we show a PQM application in which a single execution of the retrieval algorithm is sufficient to evaluate, probabilistically, the artificial neural networks architectures without the need for weights initialization. We used the PQM as a data structure to store and train artificial neural networks in superposition and to devise a quantum algorithm able to evaluate and select neural network architectures.

The objective of this work is to improve the PQM model in pattern classification tasks extending the model described in [25]. To accomplish this aim, we propose two approaches:

1. We introduce the Parametric PQM (P-PQM), where the parameter allows the PQM to compute a weighted Hamming distance and adapt the model to the given training dataset.
2. We also propose a hybrid classical-quantum version of the PQM retrieval algorithm suitable for NISQ computers.

The remainder of this work is structured as follows. Section 2 gives the concepts of quantum computing to make this work self-contained. Section 3 describes related works and the probabilistic quantum memory, and Section 4 presents a quantum weightless classifier and one of its limitations. Section 5 and Section 6 show the main contributions of this work. In Section 5, we describe the proposed parametric probabilistic quantum memory and present simulated experiments to evaluate a weightless neural network based on the P-PQM. In Section 6, we present a modification of the PQM that allows its implementation in a NISQ computer. Finally, in Section 7 we draw some conclusions and list some possible further works.

2. Quantum Computing

Quantum computing is a field that is receiving increasing attention due to its current advances [5]. It is a field that uses concepts and results from quantum mechanics and computing theory. A quantum computer is a computational device capable of representing and manipulating information at the quantum level to perform computational tasks [26]. In quantum computing, the quantum bit (qubit) represents the basic unit of information, representing a two-level quantum system. Analogously to the behavior of a subatomic particle, the qubit can be in more than one *basis* state at a given time. Eq. (1) describes one qubit in superposition (*linear combination*), where $|0\rangle$ and $|1\rangle$ are orthonormal vectors (basis), α and β are the probabilistic amplitudes (*complex numbers*) associated with the states $|0\rangle$ and $|1\rangle$, and $|\alpha|^2 + |\beta|^2 = 1$.

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

As is customary in quantum computing, the basis is fixed to $\{|0\rangle, |1\rangle\}$, and is called the *computational basis*, where

$$|0\rangle = \begin{bmatrix}1\\0\end{bmatrix} \text{ and } |1\rangle = \begin{bmatrix}0\\1\end{bmatrix}.$$

Quantum gates are used to modify the state of a quantum system.

In the computational basis, a gate is represented by a unitary matrix and carries out a reversible operation on the quantum state.

The Pauli gates are examples of quantum operators. They are defined by the matrices below.

$$X = \begin{bmatrix}0 & 1\\1 & 0\end{bmatrix}; Y = \begin{bmatrix}0 & i\\-i & 0\end{bmatrix}; Z = \begin{bmatrix}1 & 0\\0 & -1\end{bmatrix}$$

The Hadamard gate H (Eq. (2)) can create a state superposition; where $H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$.

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{2}$$

Controlled gates operate on a *target* qubit depending on the state of another qubit used as *control*.

The CNOT gate is the controlled version of the X gate and flips target qubit if the control qubit is in the state $|1\rangle$. Eq. (3) describes the CNOT gate.

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{3}$$

An important characteristic of quantum systems is the need to measure for extracting information from a quantum state. After a measurement, the system collapses to one of its superposed basis states. Given the quantum state described in Eq. (4), the probability of finding $|i\rangle$ after a measurement is $p_i = |\alpha_i|^2$ and the state will collapse to $|i\rangle$.

$$|\psi\rangle = \sum_i \alpha_i |i\rangle \tag{4}$$

Due to the capacity of dealing with states in superposition and other incorporated quantum effects (such as *entanglement*), quantum computers provide an alternative way of computing, which presumably could be used to solve problems in a way that presumably has no counterpart in classical computing [1].

3. Related works

The first model of Quantum Associative Memory (QAM) was proposed in [18] and used a variation of Grover's algorithm. The main idea of QAM is to store patterns in superposition, allowing to store 2^n patterns using n qubits. The use of Grover's algorithm is one limitation of the first QAM that searches for exact patterns and not similar patterns. The QAM performs a search for exact patterns and not similar patterns, and the required number of quantum operators limit the use of QAM in NISQ computers. A Probabilistic Quantum Memory (PQM) was proposed in [20]. The retrieval algorithm of the PQM is not based on Grover's algorithm and searches for similar patterns instead of performing an exact search. However, the retrieval algorithm is probabilistic and depends on the distribution of patterns stored in the memory.

Quantum associative memories have been used to perform classification tasks in several works [27, 17, 23, 4, 28, 29, 30]. In [29] Grover's algorithm and a quantum associative memory based on it are used to perform classification tasks in a toy dataset representing orange and apples with 3-qubit patterns. In [23] the PQM is used to classify digits from the MNIST dataset, the PQM accuracy on the test set is only 50%. An evaluation of the PQM in benchmark classification datasets was performed in [25]. One limitation of the PQM is the lack of parameters to adjust the model to a dataset [25, 23].

3.1. Probabilistic Quantum Memories

In this section, we present the quantum memory model used to build a weightless network classifier. The Probabilistic Quantum Memory (PQM) [20, 21] is a content-addressable quantum memory. It outputs the probability of a given input pattern being stored in the memory by calculating the Hamming distance between the input pattern and all the patterns stored in the memory. It is a probabilistic model designed to recognize incomplete or noisy information. Despite being an associative model, the PQM possess a highly scalable storage capability, being able to store all the possible 2^n binary patterns of n bits. The following subsections explain the PQM storage and retrieval algorithms.

3.1.1. The Storage algorithm

The storage algorithm receives a dataset $data = \cup_{i=1}^{r}\{p^i\}$ with r patterns each with n bits, uses three quantum registers and follows Algorithm 1 to produce state $|M\rangle$ described in Eq. (5).

$$|M\rangle = \frac{1}{\sqrt{r}}\sum_{i=1}^{r}\left|p^i\right\rangle \tag{5}$$

Before being processed and stored on the memory, every pattern must be initialized in an input register $|p\rangle_n$. The memory itself is separated from the input patterns which have not yet been processed and resides in the register $|m\rangle_n$. After the end of the storage algorithm, the register $|m\rangle_n$ will be in the state $|M\rangle$; $|m\rangle_n$ refers to the memory quantum register during the construction of the uniform superposition and $|M\rangle$ is the resulting memory state. The last quantum register used is an auxiliary two-qubit register $|u\rangle_2 = |u_1u_2\rangle = |u_1\rangle \otimes |u_2\rangle$, $u_i \in \{0,1\}$, which is used to keep tabs on which patterns are already stored and which still need to be processed and written. The algorithm initial state $\left|\psi_0^1\right\rangle$ using the three registers is shown in Eq. (6).

$$\left|\psi_0^1\right\rangle = |p_1p_2\cdots p_n; u_1u_2; m_1m_2\cdots m_n\rangle \tag{6}$$

Algorithm 1: Probabilistic quantum memory storage algorithm

1 Prepare the initial state $\left|\psi_0^i\right\rangle = |0_1,\cdots,0_n;01;0_1,\cdots,0_n\rangle$
2 **foreach** $p^i \in data$ **do**
3 Load p^i into quantum register $|p\rangle_n$
4 $\left|\psi_1^i\right\rangle = \prod_{j=1}^{n} 2CNOT_{p_j^i,u_2,m_j}\left|\psi_0^i\right\rangle$
5 $\left|\psi_2^i\right\rangle = \prod_{j=1}^{n} X_{m_j}CNOT_{p_j^i,m_j}\left|\psi_1^i\right\rangle$
6 $\left|\psi_3^i\right\rangle = nCNOT_{m_1\cdots m_n,u_1}\left|\psi_2^i\right\rangle$
7 $\left|\psi_4^i\right\rangle = CS_{u_1,u_2}^{r+1-i}\left|\psi_3^i\right\rangle$
8 $\left|\psi_5^i\right\rangle = nCNOT_{m_1\cdots m_n,u_1}\left|\psi_4^i\right\rangle$
9 $\left|\psi_6^i\right\rangle = \prod_{j=1}^{n} CNOT_{p_j^i,m_j}X_{m_j}\left|\psi_5^i\right\rangle$
10 $\left|\psi_7^i\right\rangle = \prod_{j=1}^{n} 2CNOT_{p_j^i,u_2,m_j}\left|\psi_6^i\right\rangle$
11 Unload p^i from quantum register $|p\rangle_n$
12 **end**

Algorithm 1 describes the storage algorithm receiving as input a dataset with r n-bit patterns. Step 1 initializes the quantum registers $|p\rangle_n|u\rangle_2|m\rangle_n$ with the quantum state $|0\rangle_n|01\rangle|0\rangle_n$. The second qubit in $|u\rangle_2$ indicates whether a pattern has been already stored or not. In this case, a $|1\rangle$ in the second qubit of $|u\rangle_2$, *.i.e.* $|u_2\rangle = |1\rangle$, indicates that the pattern has not been stored yet.

The for loop in line 2 is repeated for each pattern p^i in *data*. Step 1 initializes the quantum register $|p\rangle_n$ with a pattern p^i from *data*. Step 4 uses n $2CNOT$ operations to make a copy of the n bits from the pattern in $|p\rangle_n$ to the respective $|m\rangle_n$ register flagged by $|u_2\rangle = |1\rangle$. The $2CNOT$ operation is equivalent to a Toffoli gate that flips the target bit if the two control bits are in state $|1\rangle$.

Step 5 applies $CNOT$ operations to registers $|p\rangle_n$ and $|m\rangle_n$ followed by an X operation to $|m\rangle_n$. This step fills with 1s all the bits in the memory register which are equal to the respective bits in register $|p\rangle_n$; this is true only for the pattern which is being currently processed.

Line 6 uses the $nCNOT$ operation, which is a generalization of the $CNOT$ gate for n bits. The operation is controlled by all the bits of $|m\rangle_n$ and is applied to the first bit of $|u\rangle_n$. Thus, if $|m\rangle_n = |1\rangle_n$ the first bit of $|u\rangle_2$ is flipped.

Step 7 adds the input pattern p^i to the memory register with uniform amplitudes. This is done by applying the CS^j gate, shown below:

$$CS^j = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \sqrt{\frac{j-1}{j}} & \frac{1}{\sqrt{j}} \\ 0 & 0 & \frac{-1}{\sqrt{j}} & \sqrt{\frac{j-1}{j}} \end{bmatrix}$$

The remaining steps apply the inverse operations to return the memory to its initial state and prepare it to receive the next pattern. The algorithm runs until all the patterns have been processed and stored on $|M\rangle$.

3.1.2. The Retrieval Algorithm

The retrieval algorithm computes the Hamming distance between the input and all the patterns superposed in the memory quantum state. It probabilistically indicates the chance of a given input pattern being in the memory based on the results of its distance distribution to the stored patterns in superposition. If the input pattern is very distant from the patterns stored in the memory, one will obtain $|1\rangle$ as a result with a significant probability. Otherwise, $|0\rangle$ would be obtained. Since the memory state is in a superposition, the retrieval algorithm can calculate the distances from input to all the patterns at once.

The PQM retrieval algorithm is described in Algorithm 2. It uses three quantum registers: $|i\rangle$, $|m\rangle$ and $|c\rangle$. The pattern size gives the size of the first two registers and $|c\rangle$ is a single qubit register. Step 1 of the algorithm loads the input pattern into the first register. The second register $|m\rangle$ is the memory that contains all the stored patterns and $|c\rangle$ is a control qubit initialized with a uniform superposition of $|0\rangle$ and $|1\rangle$. The quantum state after the first step of the algorithm can be seen in Eq. (7), where r is the number of stored patterns.

Algorithm 2: Probabilistic quantum memory retrieval algorithm

1 Load the input s pattern in the quantum register $|i\rangle$
2 $|\psi_1\rangle = \prod_{j=1}^{n} X_{m_j} CNOT_{i_j,m_j} |\psi_0\rangle$
3 $|\psi_2\rangle = \prod_{i=1}^{n} \left(CU^{-2}\right)_{c,m_i} \prod_{j=1}^{n} U_{m_j} |\psi_1\rangle$
4 $|\psi_3\rangle = H_c \prod_{j=n}^{1} CNOT_{i_j,m_j} X_{m_j} |\psi_2\rangle$
5 Measure qubit $|c\rangle$
6 **if** *c == 0* **then**
7 | Measure the memory to obtain the desired state.
8 **end**

$$|\psi_0\rangle = \frac{1}{\sqrt{2r}} \sum_{k=1}^{r} \left|i; p^k; 0\right\rangle + \frac{1}{\sqrt{2r}} \sum_{k=1}^{r} \left|i; p^k; 1\right\rangle \tag{7}$$

Step 2 sets the jth qubit in memory register to $|1\rangle$, if the jth bit of input and memory are equal; and set to $|0\rangle$, if they differ. If a pattern stored in the memory is identical to the input pattern, step 2 will set its memory state to $|1\rangle_n$, where n is the pattern size. In step 3, the operators U (described in Eq. (8)) and controlled U^{-2} are applied to the memory registers.

$$U = \begin{bmatrix} e^{(i\frac{\pi}{2n})} & 0 \\ 0 & 1 \end{bmatrix} \tag{8}$$

Step 3 is responsible for computing the Hamming distance between the input pattern s and the patterns in the memory. It computes the number of 0s in the memory register (the qubits that differ between memory

and input). When $|c\rangle$ is $|0\rangle$, step 3 calculates the number of 0s in the memory state with a positive sign and with a negative sign when $|c\rangle$ is $|1\rangle$.

Step 4 reverts the memory register to its original state by computing the inverse of step 2. Step 5 measures the quantum register $|c\rangle$. An input pattern similar to the stored patterns increases the probability of measuring $|c\rangle = 0$, and an input that is very distant to the stored patterns increases the probability of measuring $|c\rangle = 1$. The measurement probabilities can be seen in Eq. (9), where r is the number of stored patterns and $d_H(s, p^k)$ denotes the Hamming distance between input and the kth stored pattern.

$$\begin{aligned} P(|c\rangle = |0\rangle) &= \sum_{k=1}^{r} \frac{1}{r} cos^2\left(\frac{\pi}{2n} d_H(s, p^k)\right) \\ P(|c\rangle = |1\rangle) &= \sum_{k=1}^{r} \frac{1}{r} sin^2\left(\frac{\pi}{2n} d_H(s, p^k)\right) \end{aligned} \tag{9}$$

4. The Quantum Weightless Classifier

The Quantum Weightless Neural Network Classifier [31] (QWC) is composed of Probabilistic Quantum Memories acting as the network neurons. The model is devised by using an array of PQM instances capable of distance-based classification. Each PQM instance, by itself, works as a single class classifier, being responsible for the classification of just one of the classes in the dataset. The model does not demand any training in the sense that the neurons do not have to be iteratively adjusted to learn from the training patterns. The model classification algorithm and the necessary setup are detailed below.

4.1. The Setup Algorithm

The Quantum Weightless Classifier requires an initial setup algorithm in order to perform classification tasks. For a given dataset with n classes, the model has n PQMs acting as neurons. The training samples must be divided and grouped by class. For each group, a new PQM is created and used to store all the samples belonging to that group, making in total n PQM instances, one for each class.

The setup algorithm consists in storing the training samples on their respective PQM. The n PQMs together define a single classifier. Algorithm 3 describes the setup process. Once all the training samples are correctly stored, the model can perform the classification task by calling the PQM retrieval algorithm.

Algorithm 3: Probabilistic Quantum Memory Classifier Setup

```
Initialize a PQM Classifier
for each class in dataset do
    Create a new PQM and assign the class label to it
    Store the class training samples on the PQM
    Add the PQM to the PQM Classifier
end
Return the PQM Classifier
```

4.2. The Classification Algorithm

After the initialization described in Algorithm 3, the next module is the classification algorithm shown in Algorithm 4. In order to classify a new sample, the Quantum Weightless Classifier must present it to all the PQM neurons which constitute its network. Each PQM neuron performs its retrieval algorithm using the presented sample as input. Since each PQM holds the patterns of a specific class, each output will be the probability of the sample having similar features to the patterns of that specific class. Let X be a random value corresponding to the number of 1s in the probabilistic quantum memory output, the PQM neuron which outputs the smallest expected value, $E(X)$, is assumed to be the one that correctly classifies the sample.

Algorithm 4: Probabilistic Quantum Memory Classifier Classification

```
for each PQM in PQM Classifier do
    Run the PQM retrieval algorithm with input testPattern
    Calculate the expected value E(X) from the retrieval algorithm output
end
Return the label from the PQM Classifier with the smallest E(X)
```

Memory 1 (M1)	Memory 3 (M3)
$m_{11} = \|0110010101\rangle$	$m_{31} = \|1111010110\rangle$
$m_{12} = \|0101010101\rangle$	$m_{32} = \|1100010101\rangle$
$m_{13} = \|0111010001\rangle$	$m_{33} = \|1101010001\rangle$
$m_{14} = \|0011010101\rangle$	$m_{34} = \|1111100101\rangle$

Table 1: Memory configurations

4.3. QWC Numeric Example

In this section, we present a numerical example of the PQM classifier evaluation. We show one limitation of the QWC with an artificial data set. Assume the memory configuration in Table 1, and that the algorithm evaluates the input pattern $|i\rangle = |0111010101\rangle$. Patterns of memory M_j in Table 1 have Hamming distance j to the input pattern $|i\rangle$.

The dataset has two classes M_1 and M_3 and 10-bit patterns. We suppose that $|i\rangle \in M_1$. To evaluate the classifier, we need to run the PQM retrieval algorithm for memories M_1 and M_3 with input $|i\rangle$ to obtain the probability of input pattern $|i\rangle$ to be in each memory. So, we begin the algorithm obtaining the initial state in each memory:

$$\begin{aligned}\left|\psi_0^{M_1}\right\rangle = \frac{1}{\sqrt{4}}\Big(&|0111010101;0110010101;C\rangle \\ &+|0111010101;0101010101;C\rangle \\ &+|0111010101;0111010001;C\rangle \\ &+|0111010101;0011010101;C\rangle\Big)\end{aligned}$$

$$\begin{aligned}\left|\psi_0^{M_3}\right\rangle = \frac{1}{\sqrt{4}}\Big(&|0111010101;1111010101;C\rangle \\ &+|0111010101;1100010101;C\rangle \\ &+|0111010101;1101010001;C\rangle \\ &+|0111010101;1111100101;C\rangle\Big)\end{aligned}$$

Applying $|\psi_1\rangle = \prod_{j=1}^{n} X_{m_j} CNOT_{i_j m_j} |\psi_0\rangle$, we obtain:

$$\begin{aligned}\left|\psi_1^{M_1}\right\rangle = \frac{1}{\sqrt{4}}\Big(&|0111010101;1110111111;C\rangle \\ &+|0111010101;1101111111;C\rangle \\ &+|0111010101;1111111011;C\rangle \\ &+|0111010101;1011111111;C\rangle\Big)\end{aligned}$$

$$\begin{aligned}\left|\psi_1^{M_3}\right\rangle = \frac{1}{\sqrt{4}}\Big(&|0111010101;0111111100;C\rangle\\ &+|0111010101;0100111111;C\rangle\\ &+|0111010101;0101111011;C\rangle\\ &+|0111010101;0111001111;C\rangle\Big)\end{aligned}$$

If we denote $d_{M_\alpha}(i, m_{\alpha k})$ as the Hamming distance between the input pattern $|i\rangle$ and every stored pattern $m_{\alpha k}$, then $d_{M_1}(i, m_{1k}) = 1$ and $d_{M_3}(i, m_{3k}) = 3$, $\forall k = 1 \ldots 4$. After step 3 of Algorithm 2 we obtain the state.

$$\begin{aligned}\left|\psi_2^{M_1}\right\rangle = &\frac{1}{\sqrt{4}\sqrt{2}}\Big(e^{(\frac{i\pi}{2n})}|i;1110111111;0\rangle\\ &+e^{(\frac{i\pi}{2n})}|i;1101111111;0\rangle\\ &+e^{(\frac{i\pi}{2n})}|i;1111111011;0\rangle\\ &+e^{(\frac{i\pi}{2n})}|i;1011111111;0\rangle\Big)\\ +&\frac{1}{\sqrt{4}\sqrt{2}}\Big(e^{(\frac{-i\pi}{2n})}|i;1110111111;1\rangle\\ &+e^{(\frac{-i\pi}{2n})}|i;1101111111;1\rangle\\ &+e^{(\frac{-i\pi}{2n})}|i;1111111011;1\rangle\\ &+e^{(\frac{-i\pi}{2n})}|i;1011111111;1\rangle\Big).\end{aligned}$$

$$\begin{aligned}\left|\psi_2^{M_3}\right\rangle = &\frac{1}{\sqrt{4}\sqrt{2}}\Big(e^{(\frac{3i\pi}{2n})}|i;0111111100;0\rangle\\ &+e^{(\frac{3i\pi}{2n})}|i;0100111111;0\rangle\\ &+e^{(\frac{3i\pi}{2n})}|i;0101111011;0\rangle\\ &+e^{(\frac{3i\pi}{2n})}|i;0111001111;0\rangle\Big)\\ +&\frac{1}{\sqrt{4}\sqrt{2}}\Big(e^{(\frac{-3i\pi}{2n})}|i;0111111100;1\rangle\\ &+e^{(\frac{-3i\pi}{2n})}|i;0100111111;1\rangle\\ &+e^{(\frac{-3i\pi}{2n})}|i;0101111011;1\rangle\\ &+e^{(\frac{-3i\pi}{2n})}|i;0111001111;1\rangle\Big)\end{aligned}$$

Applying the step 4 of Algorithm 2, we obtain:

$$\begin{aligned}\left|\psi_3^{M_1}\right\rangle = &\frac{1}{2}\Big[\cos\big(\frac{\pi}{20}\big)|i;m_{11};0\rangle + \cos\big(\frac{\pi}{20}\big)|i;m_{12};0\rangle\\ &+\cos\big(\frac{\pi}{20}\big)|i;m_{13};0\rangle + \cos\big(\frac{\pi}{20}\big)|i;m_{14};0\rangle\Big]\\ +&\frac{1}{2}\Big[\sin\big(\frac{\pi}{20}\big)|i;m_{11};0\rangle + \sin\big(\frac{\pi}{20}\big)|i;m_{12};0\rangle\\ &+\sin\big(\frac{\pi}{20}\big)|i;m_{13};0\rangle + \sin\big(\frac{\pi}{20}\big)|i;m_{14};0\rangle\Big]\end{aligned}$$

$$\begin{aligned}\left|\psi_3^{M_3}\right\rangle = \frac{1}{2}\Big[&\cos\big(\frac{3\pi}{20}\big)|i;m_{31};0\rangle + \cos\big(\frac{3\pi}{20}\big)|i;m_{32};0\rangle \\ &+\cos\big(3\frac{\pi}{20}\big)|i;m_{33};0\rangle + \cos\big(\frac{3\pi}{20}\big)|i;m_{34};0\rangle\Big] \\ +\frac{1}{2}\Big[&\sin\big(\frac{3\pi}{20}\big)|i;m_{31};0\rangle + \sin\big(\frac{3\pi}{20}\big)|i;m_{32};0\rangle \\ &+\sin\big(\frac{3\pi}{20}\big)|i;m_{33};0\rangle + \sin\big(\frac{3\pi}{20}\big)|i;m_{34};0\rangle\Big]\end{aligned}$$

The next step measures the control qubit $|C\rangle$. Therefore, we must verify the probability amplitudes of $|0\rangle$ or $|1\rangle$. If only one memory M_j returns 0, the pattern is classified as a member of the class M_j.

$P^{M_1}(|0\rangle) = cos^2\big(\frac{\pi}{20}\big) \approx 0.9755$ and $P^{M_3}(|0\rangle) = cos^2\big(\frac{3\pi}{20}\big) \approx 0.79389$. The recognition probability of the input pattern $|i\rangle$ being recognized as a pattern of memory M_1 or M_3 is high for both memories, and then the model can perform a wrong classification.

A PQM classifier [31, 23] is not a good classifier in this situation. If we increase the size of $|i\rangle$ and $\left|p^k\right\rangle$ and if the input pattern $|i\rangle$ have a Hamming distance of 1 and 3 for all stored patterns in M_x and M_y, respectively, the probability $P^{M_x}(|0\rangle)$ and $P^{M_y}(|0\rangle)$ will tend to 1 and the algorithm would indicate that $|i\rangle$ is in both classes. In the next section, we propose a parametric PQM that allows adjusting the model to a dataset.

5. Parametric Quantum Weightless Classifier

The PQM retrieval algorithm output is a function of the Hamming distances between the input and the stored patterns. This distance is not reliable for all datasets, as we have seen in the previous section. When patterns are close to patterns from another class, there is a high probability of misclassification. In order to improve the memory output probabilities, we propose a modified version of the PQM.

We show how, for all memory size $n \in \mathbb{N}$, that it is possible to add a parameter t, with $0 < t \leq 1$, to the calculation of the probability in such a way that the distance between $P^{M_x}(|0\rangle)$ and $P^{M_y}(|0\rangle)$ will be sufficient to assure a greater probability to perform a correct classification.

The idea is that given $d_x(i, m_x) < d_x(i, m_y)$, where d_x and d_y are Hamming distances between all patterns stored in M_x and M_y, respectively, we can add a parameter t in the calculation of the probabilities and obtain values for $t \in (0, 1)$ that increase the distance between the probabilities $P^{M_x}(|0\rangle)$ and $P^{M_y}(|0\rangle)$.

We want to make the probability

$$P^{M_x}(|0\rangle) = \sum_{k=1}^{p} \frac{1}{p} \cos^2\left(\frac{d_x\pi}{2nt}\right)$$

considerably close to 1, while the probability

$$P^{M_y}(|0\rangle) = \sum_{k=1}^{p} \frac{1}{p} \cos^2\left(\frac{d_y\pi}{2nt}\right)$$

stays considerably close to zero.

A Parametric Probabilistic Quantum Memory (P-PQM) operates as the PQM, but with the addition of a scale parameter in the retrieval algorithm. We replace the U operator used in the PQM with the U' operator described below, where t is a free parameter. Note that for $t = 1$ the P-PQM is equivalent to the PQM.

$$U' = \begin{bmatrix} e^{(i\frac{\pi}{2nt})} & 0 \\ 0 & 1 \end{bmatrix} \tag{10}$$

We define the Parametric Quantum Weightless Classifier (P-QWC), as the QWC classifier with a parametric retrieval algorithm. In the next section, we revisit the example of the previous section to verify the effectiveness of the parametric approach.

5.1. Numeric Example revisited

Taking the example in Section 4.3 as into account, we can apply the parameter from $|\psi_2\rangle$, hence:

$$\begin{aligned}\left|\psi_2^{M_1}\right\rangle = \frac{1}{\sqrt{4}\sqrt{2}}\Big(&e^{(\frac{i\pi}{2nt})}|i;1110111111;0\rangle \\ &+e^{(\frac{i\pi}{2nt})}|i;1101111111;0\rangle \\ &+e^{(\frac{i\pi}{2nt})}|i;1111111011;0\rangle \\ &+e^{(\frac{i\pi}{2nt})}|i;1011111111;0\rangle\Big) \\ +\frac{1}{\sqrt{4}\sqrt{2}}\Big(&e^{(\frac{-i\pi}{2nt})}|i;1110111111;1\rangle \\ &+e^{(\frac{-i\pi}{2nt})}|i;1101111111;1\rangle \\ &+e^{(\frac{-i\pi}{2nt})}|i;1111111011;1\rangle \\ &+e^{(\frac{-i\pi}{2nt})}|i;1011111111;1\rangle\Big).\end{aligned}$$

$$\begin{aligned}\left|\psi_2^{M_3}\right\rangle = \frac{1}{\sqrt{4}\sqrt{2}}\Big(&e^{(\frac{3i\pi}{2nt})}|i;0111111100;0\rangle \\ &+e^{(\frac{3i\pi}{2nt})}|i;0100111111;0\rangle \\ &+e^{(\frac{3i\pi}{2nt})}|i;0101111011;0\rangle \\ &+e^{(\frac{3i\pi}{2nt})}|i;0111001111;0\rangle\Big) \\ +\frac{1}{\sqrt{4}\sqrt{2}}\Big(&e^{(\frac{-3i\pi}{2nt})}|i;0111111100;1\rangle \\ &+e^{(\frac{-3i\pi}{2nt})}|i;0100111111;1\rangle \\ &+e^{(\frac{-3i\pi}{2nt})}|i;0101111011;1\rangle \\ &+e^{(\frac{-3i\pi}{2nt})}|i;0111001111;1\rangle\Big)\end{aligned}$$

Applying step 4 of Algorithm 2, where $n = 10$, we have:

$$\begin{aligned}\left|\psi_3^{M_1}\right\rangle = \frac{1}{2}\Big[&\cos\big(\frac{\pi}{20t}\big)|i;m_{11};0\rangle + \cos\big(\frac{\pi}{20t}\big)|i;m_{12};0\rangle \\ &+\cos\big(\frac{\pi}{20t}\big)|i;m_{13};0\rangle + \cos\big(\frac{\pi}{20t}\big)|i;m_{14};0\rangle\Big] \\ +\frac{1}{2}\Big[&\sin\big(\frac{\pi}{20t}\big)|i;m_{11};0\rangle + \sin\big(\frac{\pi}{20t}\big)|i;m_{12};0\rangle \\ &+\sin\big(\frac{\pi}{20t}\big)|i;m_{13};0\rangle + \sin\big(\frac{\pi}{20t}\big)|i;m_{14};0\rangle\Big]\end{aligned}$$

$$\begin{aligned}\left|\psi_3^{M_3}\right\rangle = \frac{1}{2}\Big[&\cos\big(\frac{3\pi}{20t}\big)|i;m_{31};0\rangle + \cos\big(\frac{3\pi}{20t}\big)|i;m_{32};0\rangle \\ &+\cos\big(3\frac{\pi}{20t}\big)|i;m_{33};0\rangle + \cos\big(\frac{3\pi}{20t}\big)|i;m_{34};0\rangle\Big] \\ +\frac{1}{2}\Big[&\sin\big(\frac{3\pi}{20t}\big)|i;m_{31};0\rangle + \sin\big(\frac{3\pi}{20t}\big)|i;m_{32};0\rangle \\ &+\sin\big(\frac{3\pi}{20t}\big)|i;m_{33};0\rangle + \sin\big(\frac{3\pi}{20t}\big)|i;m_{34};0\rangle\Big]\end{aligned}$$

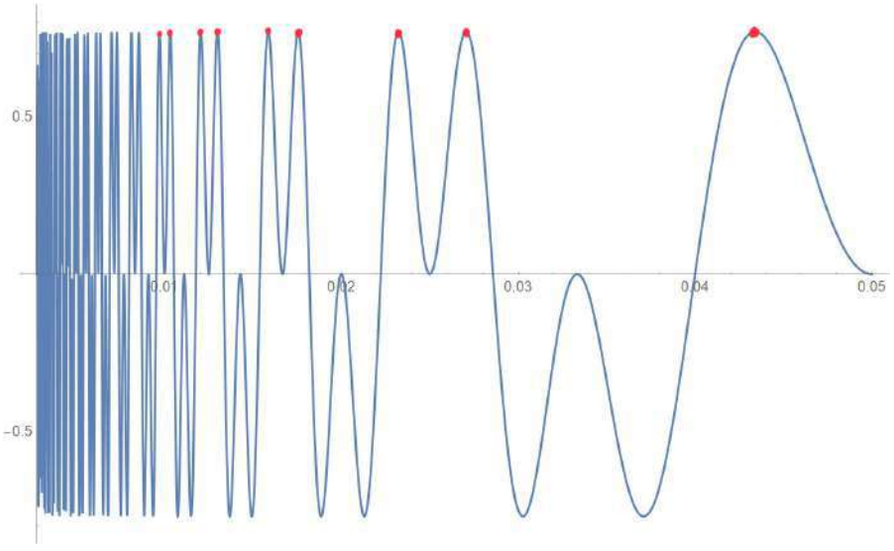


Figure 1: Graph of $f(t)$ with maximum points.

We look for a value of t which improves the probability of $|i\rangle$ to be recognized in memory M_1 and not recognized in M_3.

As $P^{M_1}(|0\rangle) = \cos^2\left(\frac{\pi}{20t}\right)$ and $P^{M_3}(|0\rangle) = \cos^2\left(\frac{3\pi}{20t}\right)$.

Let $f(t)$ be the function described in Eq. (11), where the first term is the one with the shortest Hamming distance to the input.

$$f(t) = \cos^2\left(\frac{\pi}{20t}\right) - \cos^2\left(\frac{3\pi}{20t}\right). \tag{11}$$

Any value of t that maximizes $f(t)$ is a parameter that makes the pattern recognition more reliable. It is sufficient to analyze the derivative of $f(t)$ to maximize f. By looking ath the maximum of f we can get infinite values for t that maximize the function, these values are those that give us the most significant distances between the probabilities. Below we will give some values of t that maximizes $f(t)$. Values of t where f assumes minimum values lead to the classification of the input pattern $|i\rangle$ by the memory with the greatest Hamming distance. The graph of $f(t)$ is presented in Fig. 1.

We can obtain these values of t by calculating the derivative of f. For our numeric example, we obtain $t \approx \frac{0{,}0785398}{-0.238829+3.14158n}$, $t \approx \frac{0{,}0785398}{-0.133197+3.14158n}$, $t \approx \frac{0{,}0785398}{0.238829+3.14158n}$, or $t \approx \frac{0{,}0785398}{0.133197+3.14158n}, \forall n \in \mathbb{Z}$.

With $n = 1$, we get values $t \approx 0.044$, $t \approx 0.017$, and $t \approx 0.0268$. Now let us calculate the probabilities for each value of t.

i. $P^{M_1}(|0\rangle) = \cos^2\left(\frac{\pi}{20\cdot 0.044}\right) \approx 0.83$.

i. $P^{M_3}(|0\rangle) = \cos^2\left(\frac{3\pi}{20\cdot 0.044}\right) \approx 0.079$.

ii. $P^{M_1}(|0\rangle) = \cos^2\left(\frac{\pi}{20\cdot 0.017}\right) \approx 0.96$.

ii. $P^{M_3}(|0\rangle) = \cos^2\left(\frac{3\pi}{20\cdot 0.017}\right) \approx 0.084$.

iii. $P^{M_1}(|0\rangle) = \cos^2\left(\frac{\pi}{20\cdot 0.0268}\right) \approx 0.86$.

iii. $P^{M_3}(|0\rangle) = \cos^2\left(\frac{3\pi}{20\cdot 0.0268}\right) \approx 0.024$.

Note that in all cases the distance between the memories is considerably large, giving us greater confidence that the input pattern $|i\rangle$ is recognized by the memory M_1 since the Hamming distance associated with it is the smallest.

Dataset	Classes	Instances	Attributes	Missing Values
Balance scale	3	625	4	No
Breast cancer	2	286	9	Yes
Lymphography	4	148	18	No
Mushroom	2	8124	22	Yes
SPECT Heart	2	267	22	No
Tic-tac-toe	2	958	9	No
Voting records	2	435	16	Yes
Zoo	7	101	17	No

Table 2: Datasets characteristics

We have just shown that it is possible to obtain a parameter to improve the capabilities of the PQM classifier. Even in the case where the probabilities are very close and with a high value of n, the P-QWC can solve the cases where the QWC may not be efficient. Therefore, the P-QWC classifier provides a significant improvement over the QWC. In the next section, we use benchmark datasets to compare the QWC with the proposed P-QWC.

5.2. Experiments

We present in this section the experiments conducted on a conventional computer with a reduced classical version of Algorithm 2. To simulate the Probabilistic Quantum Memory classically we followed the description of its output probability described in Eq. (9). Once the PQM classical representation is obtained, the QWC can be evaluated by following the setup and the classification algorithms described in Section 4. The same algorithm applies to the P-QWC model, with the only modification being the addition of a parameter to the Hamming distance calculation step.

To perform the experiments, we used categorical and numerical datasets from the UCI Machine Learning Repository [32]. Table 2 presents the description of the selected datasets. All datasets were preprocessed to binarize feature values and deal with any missing values. Datasets containing real numerical values were not considered. We replaced missing values by the value with the highest number of occurrences for the corresponding feature.

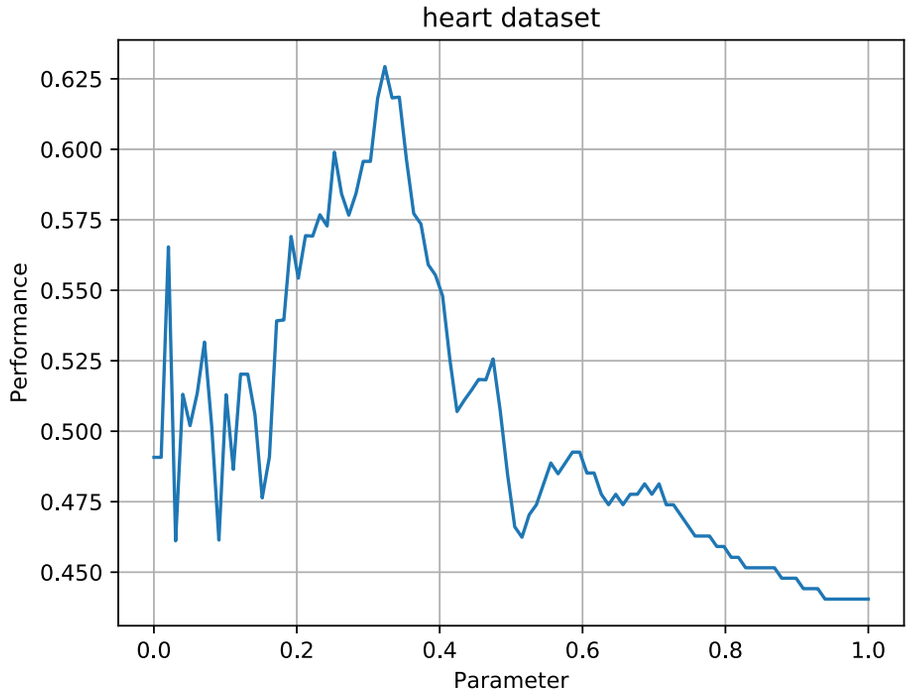
Following Algorithm 3, we stored the training samples in specific PQMs according to the classes they belong to. Then, we follow Algorithm 4. The model classification accuracy was evaluated by passing the patterns in the test set as input to each of the PQMs. The class of the PQM which outputs the lowest expected value was set as the evaluated pattern class. This algorithm was repeatedly applied to each of the evaluated datasets. For the P-QWC, model we optimized and selected the parameters which achieved the best test set accuracy for each P-PQM in the classifier. We tested 15 parameter values in the range $(0, 1]$ for each P-PQM. In a quantum computer, it would be necessary to perform several runs to estimate the best value of t. In this simulated experiment, we performed only one run and analyzed the output state to determine t.

5.3. Results

We verified that the P-QWC model performed better than the QWC over all the datasets. The parameter influence on the SPECT Heart dataset performance can be seen in Fig. 2. The curve shows the performance obtained by using the same parameter value for all P-PQMs constituting the classifier. Considering the original PQM performance is equivalent to P-PQM with parameter 1.0, a considerable increase in classification performance was observed through parameter variation and even better accuracies are possible by choosing different parameters for each P-PQM.

Table 3 describes the results obtained with the experimental setup described in Section 5.2. The accuracy of the QWC and P-QWC models can be compared against the results obtained using the k-nearest neighbors algorithm (KNN). The accuracy values shown are the average obtained from 10-fold cross-validation. Standard deviations are included between parentheses. We choose KNN as a baseline comparison because

Figure 2: Parameter impact on SPECT Heart dataset



it is also based on a measurement of distance. The KNN model was set to use uniform weights for all its points and the k nearest neighbors value was optimized, and selected from values between 1 and 50.

A nonparametric statistical test was employed to perform an appropriate comparison of the models. We used the Wilcoxon paired signed-rank test [33] with $\alpha = 0.05$ to verify whether there exist significant differences between the compared classifiers performances over the chosen datasets. Statistically significant results between the P-QWC and KNN models are marked in bold, significant results in the comparison between the QWC and P-QWC models are italicized. KNN and P-QWC are statistically equivalent in Balance scale, Breast cancer, SPECT Heart, Tic-tac-toe and Zoo datasets. KNN has better accuracy on Mushroom and Voting records datasets. P-QWC performed better on Lymphography dataset and also surpassed the QWC model in the SPECT Heart, Tic-tac-toe and Zoo datasets. In comparison with QWC, P-QWC has better accuracy in all datasets with statistically significant results in datasets SPECT Heart, Tic-tac-toe and Zoo.

The P-QWC has a performance equivalent to KNN in five out of the eight tested datasets and outperforms it in one dataset. The main advantage of the P-QWC is its memory requirements and the ability to receive inputs in superposition. While a RAM node memory grows exponentially with input size [34], the QWC memory size grows linearly. This memory advantage enables us to implement new kinds of weightless neural networks architectures [31].

6. PQM for NISQ computers

In this section, we first discuss why implementing the memory as proposed in [20] would not be feasible on noisy small-scale quantum computers. Then, we propose a hybrid classical/quantum protocol implementation, optimized for devices with a reduced number of qubits. We performed an experiment on a small-scale quantum computer as a proof of concept that the modified retrieval algorithm will work on NISQ computers.

Dataset	QWC	P-QWC	KNN
Balance scale	0.8111 (0.0666)	0.87 (0.2512)	0.8834 (0.0488)
Breast cancer	0.7309 (0.2639)	0.7380 (0.2604)	0.6970 (0.3797)
Lymphography	0.7829 (0.0815)	**0.8442** (0.1118)	0.7695 (0.0931)
Mushroom	0.886 (0.0919)	0.929 (0.073)	**1.0** (0.0)
SPECT Heart	0.4405 (0.2694)	*0.8157* (0.1113)	0.7921 (0.181)
Tic-tac-toe	0.4542 (0.1199)	*0.8309* (0.0678)	0.6714 (0.2989)
Voting records	0.892 (0.0575)	0.8966 (0.0545)	**0.9332** (0.0377)
Zoo	0.92 (0.0872)	*0.98* (0.06)	0.96 (0.0663)

Table 3: 10-fold cross-validation average accuracy per dataset

6.1. Quantum-only implementation viability analysis

The PQM retrieval algorithm calculates the distance of all the stored patterns to the input and outputs $|0\rangle$ with probability proportional to the given input pattern being close to patterns in the memory. For patterns with n bits, three quantum registers are needed for the quantum circuit: an input quantum register $|i\rangle$, a memory quantum register $|m\rangle$, both with n qubits; and a controlled quantum register with at least one qubit. The retrieval algorithm can be described in 5 steps. There are $2n$ $CNOT$ and NOT operations; n U and controlled U^{-2} operations; one Hadamard operator; and no operations involving more than two qubits. Operators U and CU^{-2} are not included in the set of gates available on the quantum device but can be constructed as a three gate composition. Thus, the significant issues with a quantum-only implementation are the memory scalability and the number of operations needed. A quantum memory used to store n-bit patterns will require $2n + 2$ qubits, for the storage algorithm; and $2n + 1$, for the retrieval algorithm. As the number of qubits is a limited resource on such small devices, the qubits requirement becomes a prohibiting issue. Furthermore, the high number of quantum operators results in too much noise being added to the output. These two issues make a quantum-only implementation on NISQ computers impractical. Therefore, due to the discussed limitations, we devised quantum-classical storage and retrieving algorithms to implement the PQM.

6.2. Hybrid quantum-classical implementation method

In the probabilistic quantum memory retrieval algorithm, the quantum register input always remains in a classical state. This fact and the hybrid classical/quantum architecture used in the actual quantum computers allows the removal of the input quantum register from Algorithm 2.

The main disadvantage of this approach is the need to recompile the circuit when the system receives an input, but as we show in [4] the PQM has applications with a fixed input. Here, we remove the input quantum register and keep it in a classical variable. All the control operators from the input register to memory register are removed from the circuit and replaced with an X operation applied to the jth memory bit only when the jth input bit is 1. We also remove the X gates from steps 2 and 4, as they would cancel with the X gates used to obtain $U = XCu_1X$. With these modifications, we can use the 5-qubit quantum computer to simulate a probabilistic quantum memory up to 4-bit patterns.

In this work, we used the Quantum Information Software Kit (QISKit) SDK [35] and run the circuit on the IBM Q Experience "Tenerife" 5-qubit quantum computer [36].

The memory retrieval algorithm operations are further simplified by breaking down complex quantum operations into classical-quantum equivalent algorithms. These algorithms are conditioned on the classical input and apply a smaller number of gates to achieve the same resulting state.

6.3. Experiments

The Tenerife 5-qubit quantum computer architecture does not have arbitrary qubits connections. All the possible connections in the Tenerife computer are defined by its architecture topology, which can be seen in Fig. 3.

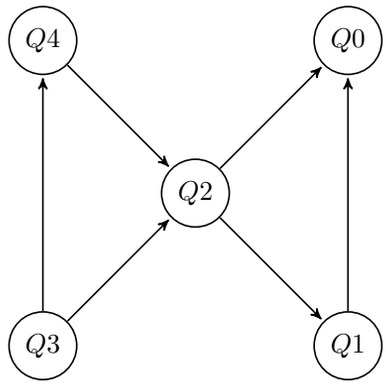


Figure 3: IBM Q Experience "Tenerife" 5-qubit quantum computer topology

For this experiment, the PQM is tested with 2, 3 and 4 qubits memory size. We store different patterns on the PQM and run the retrieval algorithm using inputs with the same size as the memory patterns size. The PQMs are constructed by storing the control bit c on the quantum register labelled as $Q3$ on Tenerife's topology. In this way, the necessary gates could be directly applied without swapping qubits. Table 4 describes the results after 8192 executions of the probabilistic quantum memory running on the Tenerife backend and the local QISKit simulator. The calculated expected values for each memory state are included as well.

We calculate the expected outputs, obtained through numerical evaluation; the real outputs, obtained from the Tenerife backend; and the simulation outputs from the QISKit simulator. We calculate the percentual Mean Squared Error (MSE) of the expected outputs and the outputs of the experiment on the Tenerife backend. In the first set of experiments, we use a PQM with memory size equal to one and present all possible binary inputs. With memory state $|0\rangle$, $|1\rangle$ and $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, we obtain, respectively, MSE equal to 0.0058, 0.0059 and 0.0006. The resulting mean error was 0.0041.

In the second set of experiments, we use a 2-qubit PQM, we use all possible 2-bit strings as inputs. With memory states $|00\rangle$, $\frac{1}{\sqrt{2}}(|00\rangle + |01\rangle)$ and $|11\rangle$, we obtain, respectively, MSE equal to 0.0072, 0.0027 and 0.0066. The resulting mean error was 0.0055.

In the third set of experiments, we use a 3-qubit PQM and all possible 3-bit strings as inputs. With memory states $|000\rangle$, $\frac{1}{\sqrt{2}}(|000\rangle + |010\rangle)$, $\frac{1}{\sqrt{2}}(|000\rangle + |100\rangle)$, $\frac{1}{\sqrt{2}}(|000\rangle + |001\rangle)$, $\frac{1}{\sqrt{2}}(|110\rangle + |111\rangle)$ and $|111\rangle$, we obtain, respectively, MSE equal to 0.0257, 0.0187, 0.0133, 0.023, 0.0218, and 0.026. The resulting mean error was 0.0214.

Finally, in the fourth set of experiments we use a 4-qubit PQM and all possible 4-bit strings as inputs. With memory states $|0000\rangle$, $\frac{1}{\sqrt{2}}(|0000\rangle + |0100\rangle)$, $|1000\rangle$, $\frac{1}{\sqrt{2}}(|0100\rangle + |1100\rangle)$, $|1010\rangle$, $\frac{1}{\sqrt{2}}(|0110\rangle + |1110\rangle)$, $|1110\rangle$, $\frac{1}{\sqrt{2}}(|0111\rangle + |1111\rangle)$ and $|1111\rangle$, we obtain, respectively, MSE equal to 0.0228, 0.0174, 0.0236, 0.0144, 0.0228, 0.0148, 0.0248, 0.0141, and 0.0226. The resulting mean error was 0.0197. Results for inputs 0000 and 1111 are displayed in Fig. 4.

Although the calculated outputs do not correspond precisely to the predicted outputs, the probability of obtaining $|0\rangle$ from a measurement of the quantum register $|c\rangle$ is still related to the distance between the input and the memory content. In all experiments, the estimate of $|c\rangle$ can be used to verify if a pattern is close to the patterns in the memory.

The proposed simplified retrieval algorithm of the PQM was successfully implemented in the Tenerife architecture without swapping quantum bits. We conjecture that a quantum computer with planar architecture where n qubits are connected to a single qubit (to be used as a quantum register $|c\rangle$) can be used to efficiently implement an n-qubits PQM in near term quantum computers.

Table 4: Probabilistic quantum memory retrieval algorithm for input $|0\rangle_{size}$ executed on the Tenerife backend with 8192 executions. The results for 1, 2, 3, and 4-qubit PQM can be seen for different memory configurations. The table shows the real output obtained from the quantum device, the results from the local simulator, and the expected output probability calculated numerically. We denote by Ps the Pattern Size, and by P_T, P_{Ls} and P_{Ep} the probabilities obtained from the Tenerife backend, the local QISKit simulator and the calculated expected output, respectively.

Ps	Memory state	$P_T(\|c\rangle = \|0\rangle)$	$P_{Ls}(\|c\rangle = \|0\rangle)$	$P_{Ep}(\|c\rangle = \|0\rangle)$
1	$\|0\rangle$	0.9374	1.0000	1.0000
1	$\frac{1}{\sqrt{2}}(\|0\rangle + \|1\rangle)$	0.5095	0.4974	0.5000
1	$\|1\rangle$	0.0913	0.0000	0.0000
2	$\|00\rangle$	0.9033	1.0000	1.0000
2	$\frac{1}{\sqrt{2}}(\|00\rangle + \|01\rangle)$	0.6854	0.7438	0.7500
2	$\|11\rangle$	0.1224	0.0000	0.0000
3	$\|000\rangle$	0.7618	1.0000	1.0000
3	$\frac{1}{\sqrt{2}}(\|000\rangle + \|010\rangle)$	0.6758	0.8782	0.8750
3	$\frac{1}{\sqrt{2}}(\|000\rangle + \|100\rangle)$	0.6924	0.8735	0.8750
3	$\frac{1}{\sqrt{2}}(\|000\rangle + \|001\rangle)$	0.6246	0.8804	0.8750
3	$\frac{1}{\sqrt{2}}(\|110\rangle + \|111\rangle)$	0.2925	0.1257	0.125
3	$\|111\rangle$	0.2278	0.0000	0.0000
4	$\|0000\rangle$	0.7545	1.0000	1. 0000
4	$\frac{1}{\sqrt{2}}(\|0000\rangle + \|0100\rangle)$	0.7432	0.9271	0.9268
4	$\|1000\rangle$	0.7303	0.856	0.8535
4	$\frac{1}{\sqrt{2}}(\|0100\rangle + \|1100\rangle)$	0.6844	0.67	0.6768
4	$\|1010\rangle$	0.5441	0.4961	0.5
4	$\frac{1}{\sqrt{2}}(\|0110\rangle + \|1110\rangle)$	0.4830	0.3336	0.3232
4	$\|1110\rangle$	0.3827	0.1396	0.1465
4	$\frac{1}{\sqrt{2}}(\|0111\rangle + \|1111\rangle)$	0.2423	0.073	0.0732
4	$\|1111\rangle$	0.2203	0.0000	0.0000

7. Conclusion

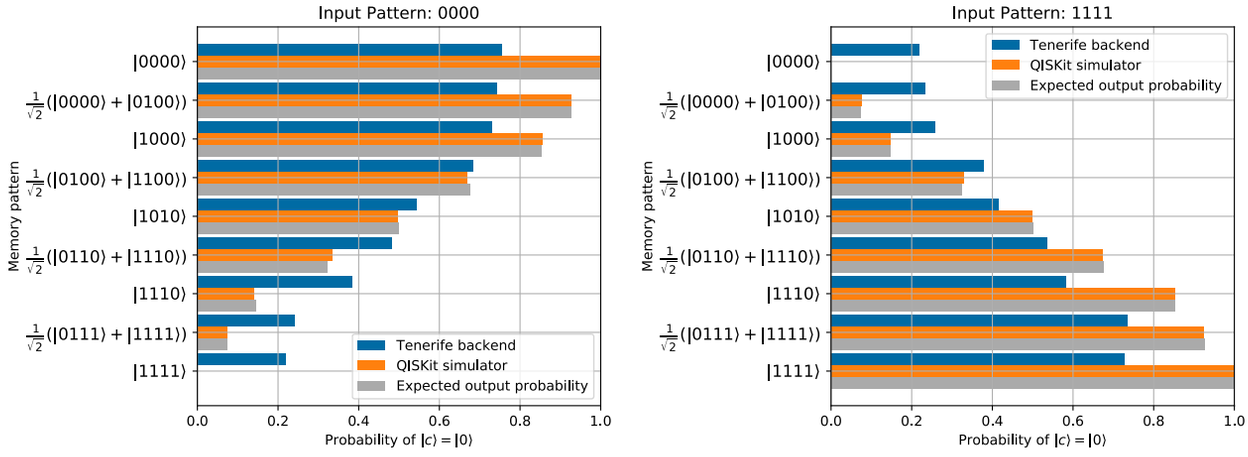
In this work, we proposed a parametric version of the PQM named P-PQM. We performed an empirical evaluation of a quantum weightless classifier and proposed a modification which achieved a considerable improvement in the classification capabilities of the model. The proposed parametric quantum model used as a classifier (P-QWC) performed better or equivalent to its unmodified version (QWC) in all datasets used in the experiments.

We also presented a modification of the PQM to allow its implementation on Noisy Intermediate-Scale Quantum computers. As a proof of concept, the model was implemented in a small-scale quantum computer. We verified through experiments that a noisy version of the PQM can be implemented on a 5-qubit quantum computer.

There are several possible future works. We can adapt the model to use different distance functions, use different architectures for the quantum weightless classifier, investigate how to reduce noise on the P-PQM, and define an error cost function to update the parameter t as in a variational quantum circuit.

Code avaiability

All code used in this work can be made available upon reasonable request.



(a) Results for input pattern 0000

(b) Results for input pattern 1111

Figure 4: Results obtained from executing the retrieval algorithm of a 4-qubit PQM on the Tenerife backend and the QISKit local simulator, including the numerically calculated expected probabilities.

Acknowledgements

This work was supported by the Serrapilheira Institute (grant number Serra-1709-22626), CNPq Edital Universal (grants numbers 420319/2016-6 and 421849/2016-9 and FACEPE (grant number IBPG-1578-1.03/16). We acknowledge use of the IBM Q for this work. The views expressed are those of the authors and do not reflect the official policy or position of IBM or the IBM Q team.

References

[1] P. W. Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, SIAM review 41 (2) (1999) 303–332.
[2] L. K. Grover, Quantum mechanics helps in searching for a needle in a haystack, Physical review letters 79 (2) (1997) 325.
[3] P. Rebentrost, M. Mohseni, S. Lloyd, Quantum support vector machine for big data classification, Physical review letters 113 (13) (2014) 130503.
[4] P. G. dos Santos, R. S. Sousa, I. C. Araujo, A. J. da Silva, Quantum enhanced cross-validation for near-optimal neural networks architecture selection, International Journal of Quantum Information (2018) 1840005.
[5] J. Preskill, Quantum computing in the NISQ era and beyond, arXiv preprint arXiv:1801.00862 (2018).
[6] A. W. Harrow, A. Montanaro, Quantum computational supremacy, Nature 549 (7671) (2017) 203.
[7] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, S. Lloyd, Quantum machine learning, Nature 549 (7671) (2017) 195.
[8] T. Mitchell, Machine Learning, McGraw-Hill, 1997.
[9] C. Shi, C.-M. Pun, Adaptive multi-scale deep neural networks with perceptual loss for panchromatic and multispectral images classification, Information Sciences 490 (2019) 1 – 17.
[10] H. Zhang, Q. Zhang, L. Ma, Z. Zhang, Y. Liu, A hybrid ant colony optimization algorithm for a multi-objective vehicle routing problem with flexible time windows, Information Sciences 490 (2019) 166 – 190.
[11] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, J. M. Gambetta, Supervised learning with quantum-enhanced feature spaces, Nature 567 (7747) (2019) 209.
[12] A. J. da Silva, T. B. Ludermir, W. R. de Oliveira, Quantum perceptron over a field and neural network architecture selection in a quantum computer, Neural Networks 76 (2016) 55–64.
[13] A. Fawaz, P. Klein, S. Piat, S. Severini, P. Mountney, Training and meta-training binary neural networks with quantum computing, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, ACM, 2019, pp. 1674–1681.
[14] V. Dunjko, J. M. Taylor, H. J. Briegel, Quantum-enhanced machine learning, Physical review letters 117 (13) (2016) 130501.
[15] A. W. Harrow, A. Hassidim, S. Lloyd, Quantum algorithm for linear systems of equations, Physical review letters 103 (15) (2009) 150502.

[16] K. H. Wan, O. Dahlsten, H. Kristjánsson, R. Gardner, M. Kim, Quantum generalisation of feedforward neural networks, npj Quantum Information 3 (1) (2017) 36.
[17] M. Schuld, M. Fingerhuth, F. Petruccione, Implementing a distance-based classifier with a quantum interference circuit, EPL (Europhysics Letters) 119 (6) (2017) 60002.
[18] D. Ventura, T. Martinez, Quantum associative memory, Information Sciences 124 (1-4) (2000) 273–296.
[19] A. Ezhov, A. Nifanova, D. Ventura, Quantum associative memory with distributed queries, Information Sciences 128 (3-4) (2000) 271–293.
[20] C. A. Trugenberger, Probabilistic quantum memories, Physical Review Letters 87 (6) (2001) 067901.
[21] C. A. Trugenberger, Quantum pattern recognition, Quantum Information Processing 1 (6) (2002) 471–493.
[22] T. Brun, H. Klauck, A. Nayak, M. Rötteler, C. Zalka, Comment on "probabilistic quantum memories", Phys. Rev. Lett. 91 (2003) 209801. doi:10.1103/PhysRevLett.91.209801.
URL https://link.aps.org/doi/10.1103/PhysRevLett.91.209801
[23] M. Schuld, I. Sinayskiy, F. Petruccione, Quantum computing for pattern classification, in: Pacific Rim International Conference on Artificial Intelligence, Springer, 2014, pp. 208–220.
[24] V. Dunjko, H. J. Briegel, Machine learning & artificial intelligence in the quantum domain: a review of recent progress, Reports on Progress in Physics 81 (7) (2018) 074001.
[25] P. G. dos Santos, R. S. Sousa, A. J. da Silva, A wnn model based on probabilistic quantum memories, in: European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning 2019 Proceedings, 2019, pp. 313–318.
[26] M. A. Nielsen, I. Chuang, Quantum computation and quantum information, AAPT, 2002.
[27] R. Zhou, Quantum competitive neural network, International Journal of Theoretical Physics 49 (1) (2010) 110.
[28] F. M. de Paula Neto, A. J. da Silva, W. R. de Oliveira, T. B. Ludermir, Quantum probabilistic associative memory architecture, Neurocomputing 351 (2019) 101–110.
[29] M. P. Singh, K. Radhey, V. Saraswat, S. Kumar, Classification of patterns representing apples and oranges in three-qubit system, Quantum Information Processing 16 (1) (2017) 16.
[30] J.-P. T. Njafa, S. N. Engo, Quantum associative memory with linear and non-linear algorithms for the diagnosis of some tropical diseases, Neural Networks 97 (2018) 1–10.
[31] A. Silva, W. de Oliveira, T. Ludermir, A weightless neural node based on a probabilistic quantum memory, in: 2010 Eleventh Brazilian Symposium on Neural Networks, IEEE, 2010, pp. 259–264.
[32] D. Dheeru, E. Karra Taniskidou, UCI machine learning repository (2017).
URL http://archive.ics.uci.edu/ml
[33] J. Demšar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine learning research 7 (Jan) (2006) 1–30.
[34] I. Aleksander, M. De Gregorio, F. M. G. França, P. M. V. Lima, H. Morton, A brief introduction to weightless neural systems., in: ESANN, Citeseer, 2009, pp. 299–305.
[35] G. Aleksandrowicz, T. Alexander, P. Barkoutsos, L. Bello, Y. Ben-Haim, D. Bucher, F. J. Cabrera-Hernández, J. Carballo-Franquis, A. Chen, C.-F. Chen, J. M. Chow, A. D. Córcoles-Gonzales, A. J. Cross, A. Cross, J. Cruz-Benito, C. Culver, S. D. L. P. González, E. D. L. Torre, D. Ding, E. Dumitrescu, I. Duran, P. Eendebak, M. Everitt, I. F. Sertage, A. Frisch, A. Fuhrer, J. Gambetta, B. G. Gago, J. Gomez-Mosquera, D. Greenberg, I. Hamamura, V. Havlicek, J. Hellmers, Łukasz Herok, H. Horii, S. Hu, T. Imamichi, T. Itoko, A. Javadi-Abhari, N. Kanazawa, A. Karazeev, K. Krsulich, P. Liu, Y. Luh, Y. Maeng, M. Marques, F. J. Martín-Fernández, D. T. McClure, D. McKay, S. Meesala, A. Mezzacapo, N. Moll, D. M. Rodríguez, G. Nannicini, P. Nation, P. Ollitrault, L. J. O'Riordan, H. Paik, J. Pérez, A. Phan, M. Pistoia, V. Prutyanov, M. Reuter, J. Rice, A. R. Davila, R. H. P. Rudy, M. Ryu, N. Sathaye, C. Schnabel, E. Schoute, K. Setia, Y. Shi, A. Silva, Y. Siraichi, S. Sivarajah, J. A. Smolin, M. Soeken, H. Takahashi, I. Tavernelli, C. Taylor, P. Taylour, K. Trabing, M. Treinish, W. Turner, D. Vogt-Lee, C. Vuillot, J. A. Wildstrom, J. Wilson, E. Winston, C. Wood, S. Wood, S. Wörner, I. Y. Akhalwaya, C. Zoufal, Qiskit: An Open-source Framework for Quantum Computing (Jan. 2019). doi:10.5281/zenodo.2562111.
URL https://doi.org/10.5281/zenodo.2562111
[36] I. Q. team, IBM Q 5 Tenerife backend specification v1.3.0, https://ibm.biz/qiskit-tenerife, online; accessed 24-September-2018 (2018).