POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

Design of a Quantum Circuit for Quantum Random Walks on Johnson Graphs

Solving the ISD problem for Code-based Post-quantum Cryptosystems

Laurea Magistrale in Computer Engineering - Ingegneria Informatica

Author: Giacomo Lancellotti, Matteo Lodi

Advisor: Prof. Alessandro Barenghi

Co-advisors: Prof. Gerardo Pelosi, Simone Perriello

Academic year: 2020-2021

---

1. Introduction

Quantum computers have drastically risen in popularity mainly due to big investments from both public and private institutions. The main reason behind this technological break-through is the possibility of solving very specific problems in a much more efficient way compared to classical methods of computation.
Cryptography is one of the mathematical fields influenced the most by such advantage, which is the reason why today's literature focuses on a category of cryptoschemes defined as quantum-resistant. Among them, Code-based cryptosystems rely on the hardness of finding a minimum-weight codeword in a seemingly randomly structured linear code. The most effective attacks to such systems try to solve what's called the Information Set Decoding problem. As a consequence, any quantum procedure that could speedup its computation is of great significance. In this work we show a circuital implementation of a Quantum random walk in a Johnson graph which, to the best of our knowledge, has only been studied in literature from a pure mathematical point of view. We also prove that this solution is mathematically correct by comparing simulation results obtained from Qiskit, IBM's framework for quantum simulation, with Thomas Wong's work on Lackadaisical Quantum random walks [7]. In addition, we also analyze the circuit's performance when applied to Finasz-Sendrier's Information Set Decoding algorithm [2], indicating its gate number as well as depth using the algorithm's parameters.

2. Coding Theory

A linear code $\mathcal{C}$ can be seen as a set of vectors: $\mathcal{C} := \{\mathbf{c} \in \mathbb{F}_q^n | \mathbf{c} = G^\top \mathbf{m}, G \in \mathbb{F}_q^{k\times n}, \mathbf{m} \in \mathbb{F}_q^k\}$ where the generator matrix G allows us to encode any word $\mathbf{m}$ of length k into a codeword $\mathbf{c}$ of length n belonging to the code. By defining a parity check matrix $H \in \mathbb{F}_q^{(n-k)\times n}$ such that $HG^\top = 0_{(n-k)\times k}$ it follows that $H\mathbf{c} = HG^\top \mathbf{m} = 0$. Assume that $\mathbf{y} \in \mathbb{F}_q^n$ is a message afflicted by an error $\mathbf{e}$ of $wt(\mathbf{e}) = t$ (where $wt()$ is the Hamming weight) to be decoded $\mathbf{y} = \mathbf{e}+\mathbf{c}$, to check if $\mathbf{y}$ is error-free we can use the parity check matrix and computing $H\mathbf{y} = H\mathbf{e} + H\mathbf{c} = \mathbf{s}$, $\mathbf{s} \in \mathbb{F}_q^{(n-k)}$ is the syndrome associated to the vector $\mathbf{y}$.

Decoding a corrupted message consists of finding the proper error $\mathbf{e}$ that affects the codeword, this is known in literature as Syndrome Decoding Problem.
The idea behind almost all ISD algorithms is to guess an Information Set $\mathcal{I}^*$ that corresponds to a set of linear independent columns of H for which we know the indexed part of the error $\mathbf{e}_{\mathcal{I}^*}$ and consequently reduce the search space.
Finiasz and Sendrier algorithm (FS-ISD) [2] applies a further refinement of this concept, by having $H_{\mathcal{I}^*}$ in its systematic form $\hat{H} = [V|I_{(n-k)}]$ and consequently $\hat{H}\hat{\mathbf{e}} = [V|I_{(n-k)}][\hat{\mathbf{e}}_{\mathcal{I}}^\top|\hat{\mathbf{e}}_{\mathcal{I}^*}^\top]^\top = V\hat{\mathbf{e}}_{\mathcal{I}} + \hat{\mathbf{e}}_{\mathcal{I}^*} = \hat{\mathbf{s}}$ they also allow that part of the weight of the permuted error $\hat{\mathbf{e}}$ to be concentrated in its first $k+l$ positions $wt(\hat{\mathbf{e}}_{\mathcal{I}}) = p$. The resulting weight distribution is shown in Fig. 1.
Their strategy is to split $\hat{\mathbf{e}}_{\mathcal{I}}$ in two intervals of $(k+l)/2$ bits each of weight $p/2$. Finding the correct $\hat{\mathbf{e}}_{\mathcal{I}}$ is equal to solving the equation 1.

$$V_{up}\hat{e}_{\mathcal{I}} = \hat{s}_{up} \Leftrightarrow \hat{s}_{up} = V_{up1}\hat{e}_{\mathcal{I},up1} + V_{up2}\hat{e}_{\mathcal{I},up2} \tag{1}$$

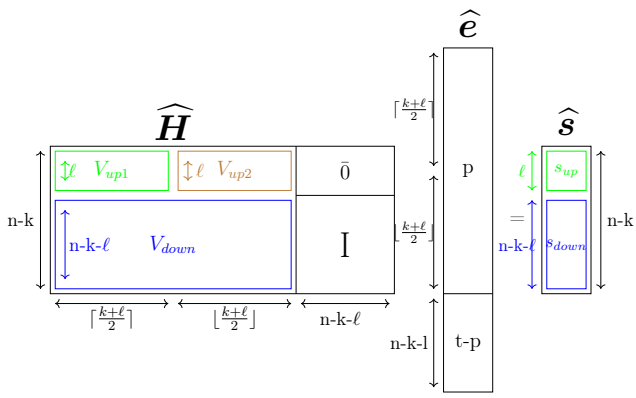

Figure 1: Weight distribution for Finiasz-Sendrier's algorithm

It can be derived that $V_{up2}\hat{\mathbf{e}}_{\mathcal{I},up2} = \hat{\mathbf{s}}_{up} + V_{up1}\hat{\mathbf{e}}_{\mathcal{I},up1}$.
Solving this equality can be reformulated as finding a collision between two sets of $\binom{k+l/2}{p/2}$ elements. Classically this is known as 2-SUM problem and it is what we are going to improve with our quantum algorithm.

3. Quantum Random Walks

The concept of Random Walk was introduced at the beginning of the last century as a simple mathematical curiosity, but it turned out to have extremely interesting properties when employed as structure for exploration/search problems. Using a Markov chain formulation, they can be described as a stochastic process in which a walker explores a set of vertices connected by edges that form a directed graph. At each time step, the walker randomly decides which adjacent vertex they explore next. The 'speed' of exploration of a graph using this technique is indicated by the standard deviation of the probability distribution of being in a generic vertex and time step, starting from the initial position (shown in Fig. 2). Classically speaking, this deviation is sub-linear w.r.t. the time passed since the start of the exploration, specifically $\sigma(t) = \sqrt{t}$.

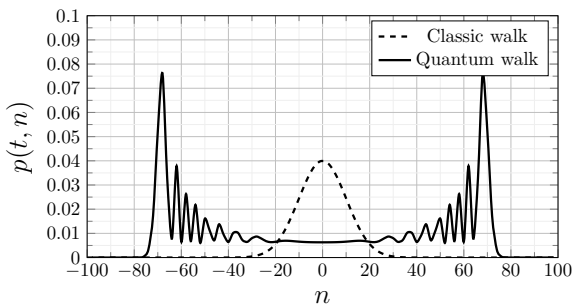

Figure 2: Comparison between a classical and a Quantum random walk (walk on a line, $t = 100$)

Quantum Random Walks, term coined by Aharonov et al. in 1993 [1], are the Quantum equivalent of random walks. Using the walk on the line as example, the walker's position is now a vector in the Hilbert space $\mathcal{H}_P$ called *state space* with computational basis $\{|n\rangle : n \in \mathcal{Z}\}$. The choice of the direction for the next walk step is a random coin toss represented by the *coin space*. In our example, it is a two-dimensional Hilbert space $\mathcal{H}_C$ with basis $\{|0\rangle, |1\rangle\}$. The complete Hilbert space is $\mathcal{H} = \mathcal{H}_C \otimes \mathcal{H}_P$.
To perform the exploration, we need to apply a sequence of transformations described by the following unitary operators:

- C coin operator: the decision maker regarding the next step direction. By acting on the coin space, it acts as a coin toss the choose where to go next.
- S shift operator: it updates the current position according to the chosen direction by modifying the state space.

One step of the exploration is thus first an application of the coin operator followed by the shift operator. We call this transformation sequence evolution operator $U = SC$.
As we can see from the plot of the quantum probability distribution in Fig. 2, this walk behaves quite differently compared to its classical counterpart. The standard deviation is now $\sigma(t) = 0.54t$, which means that on average the walker will explore at a quadratically faster speed.
Following the model proposed by Shenvi, Kempe and Whaley in [6], in order to be able to encode a search problem in this setting we also need a quantum operator that checks whether a given element respects a certain condition. We call this operator Oracle O.

$$O|x\rangle|0\rangle = \begin{cases} |x^*\rangle\,|1\rangle & \text{if } x = x^* \\ |x\rangle\,|0\rangle & \text{otherwise} \end{cases} \tag{2}$$

where x is a generic element and x^* is the marked one.
Looking at Equation (2), the first register is the state register whereas the second register is formed by ancillas that stores the output of the checked condition: $|1\rangle$ for x^* of $|0\rangle$ for the generic non-marked $|x\rangle$. This register will be used to control the coin application for the marked element. Considering the uncomputation of these ancillas, necessary for the successive exploration step, this operation modifies the evolution operator to $U = SO^{-1}CO$.
To conclude this section, Fig. 3 shows a complete circuit for a generic Quantum random walk search with a state register of n qubits and a coin register of d qubits. After an initialization of the state and coin registers, we apply $\mathcal{O}(\sqrt{N})$ times the evolution operator and apply a final measurement of the state space.

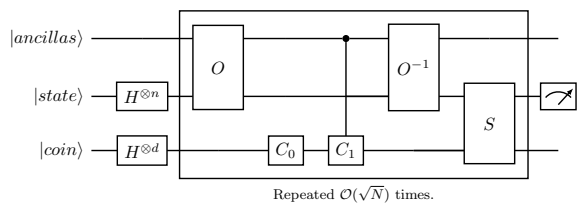

Figure 3: Generic Quantum random walk search circuit

4. Quantum Search on a Johnson Graph

Let $\mathcal{G}(V, E)$ be the Johnson graph $J(n, k)$ where V is the set of vertices and E is the set of edges. V is the set of k-subsets of $[n] = \{1, 2, \ldots, n\}$, and two vertices $v, v' \in V$ are adjacent if and only if $|v \cap v'| = k-1$. The number of vertices is $\binom{n}{k}$. A $J(n, k)$ Johnson graph is d-regular, where d is the degree equal to $d = k(n-k)$. This kind of graphs is also vertex-transitive as well as distance-transitive, meaning a search problem encoded in such structure is independent of the location of the marked vertex and that non-marked vertices can be partitioned into subsets depending on their distance from the marked vertex. These properties render Johnson graphs, as well as other regular graphs, a good way of encoding structured search problems. Fig. 4 shows the structure of a $J(4, 2)$ graph with self-loops. The binary labels in the next Figure can give an understanding of how the encoding works. Every vertex is associated with a n-bit long binary string with Hamming weight equals to k totaling $\binom{n}{k}$ vertices, and is adjacent to another vertex only if their Hamming distance is equal to 2.

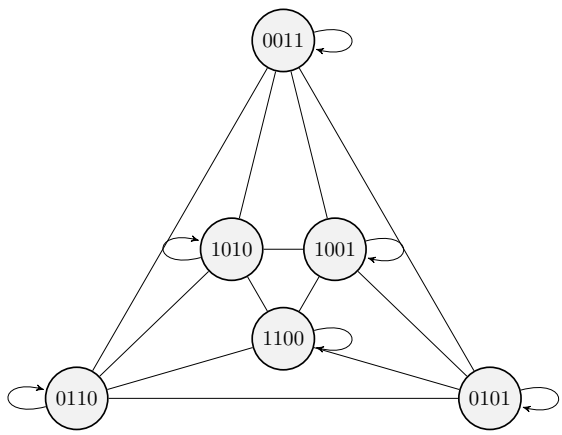

Figure 4: Johnson graph $J(4, 2)$ with binary labels and self-loops

In order to perform a Quantum walk on this structure, we need to set up a proper superposition that represents all possible vertices in a generic $J(n, k)$ Johnson graph. This is known in literature as Dicke state $|D_k^n\rangle$: it is an equal-weight superposition of all n-qubit states with Hamming weight k. The superposition is described as:

$$|D_k^n\rangle = \binom{n}{k}^{-\frac{1}{2}} \sum_{wt(x)=k} |x\rangle \qquad x \in \{0,1\}^n \tag{3}$$

In order to obtain such a superposition, we implemented in our work the circuit defined by Mukherjee et al. in [5].
Ideally, given a generic vertex, to make a step towards its neighbours one should apply a swap operation between two of the state register's qubits having opposite boolean value. With a state space represented by n qubits, out of $\binom{n}{2}$ possible swaps only $k(n-k)$ of them are 'correct' meaning they preserve the Hamming distance of 2. The problem rises when taking into consideration the fact that each vertex needs a different set of swaps in order to shift to its neighbours. An example of different swap sets for two vertices of $J(4,2)$ is shown in Fig. 5.

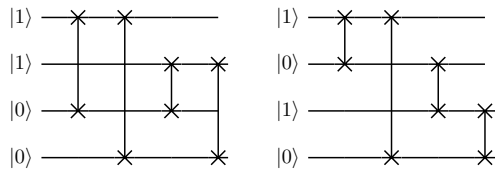

Figure 5: Different swap sets for vertices 1100 and 1010

Our solution performs all $\binom{n}{2}$ swaps. The result is that only $k(n-k)$ of those swaps will shift to a neighbour of any given vertex, whereas the rest will apply a label swap that won't meaningfully modify the starting vertex: we have essentially performed a self-loop (shown in Fig. 4).
Quantum random walks with self-loops are defined in literature as Lackadaisical Quantum random walks and have been extensively studied in [7]. They are characterized by an additional coin degree of freedom that encodes the self-loop. As an example, in a Johnson $J(4,2)$ (which has degree $d = 4$) the shifting directions encoded via the coin gate are $\{|a\rangle, |b\rangle, |c\rangle, |d\rangle, |\circlearrowleft\rangle\}$. The weight of the self-loops (number of self loops of a vertex) of our solution is equal to

$$\gamma = 2^{\lceil \log_2 \binom{n}{2} \rceil} - k(n-k) \tag{4}$$

This is explained by the fact that in order to encode in the coin register $\binom{n}{2}$ directions, we need at least $\lceil \log_2 \binom{n}{2} \rceil$ qubits. As a consequence the coin register has an expressive power of $2^{\lceil \log_2 \binom{n}{2} \rceil}$ directions. All non-meaningful ones count as self-loops. An important characteristic of this kind of walk is that this weight value (γ) influences the maximum probability of measuring the marked vertex (p^*) as well as the necessary time-steps to reach the first high probability peak (t^*). Specifically, [7] defines

$$\begin{aligned} t^* &= \frac{\pi}{\sqrt{2(\gamma+1)}} \sqrt{\binom{n}{k}} \\ p^* &= \frac{4\gamma}{(\gamma+1)^2} \end{aligned} \tag{5}$$

Our solution's weight value is tied to the Johnson's structure (Equation (4) has only n and k as parameters) and does not achieve the maximum probability $p^* = 1$. As a consequence multiple measurements of our Quantum walk are necessary. If the value of n becomes too high, the self-loop value causes the search to slow down too much, rendering it almost irrelevant.

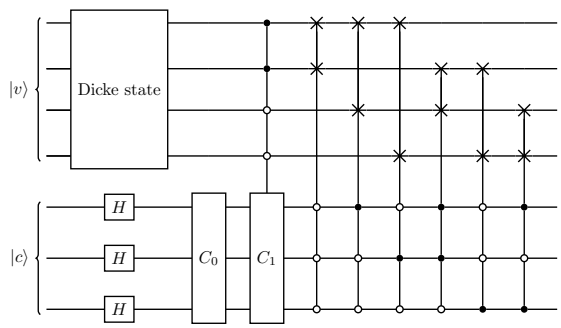

Figure 6: Complete Johnson graph walk search with marked vertex 1100(1 step)

Our final circuit that performs a Quantum random walk search on a Johnson graph is shown in Fig. 6 and requires the following qubit registers:

- $|v\rangle$: state register. It is used to encode the vertices of the graph, every qubit corresponds to a position in the set n thus it has dimension $dim(|v\rangle) = n$
- $|c\rangle$: coin register. It expresses all the possible directions of the coin. It has dimension $dim(|c\rangle) = \lceil \log_2 \binom{n}{2} \rceil$

To sum up, the algorithms phases are:

- **Initialization phase**: Dicke state preparation on the state register, Hadamard gates on the coin register.
- **Coin phase**: Grover operator on the coin register followed by a controlled re-application of the same operator.
- **Shift phase**: $\binom{n}{2}$ swap operations of the state register controlled by the coin register.

The Coin and Shift phases are then repeated $\mathcal{O}(\binom{n}{k})$ times.
We simulated the circuit shown in Fig. 6 obtaining the following curve

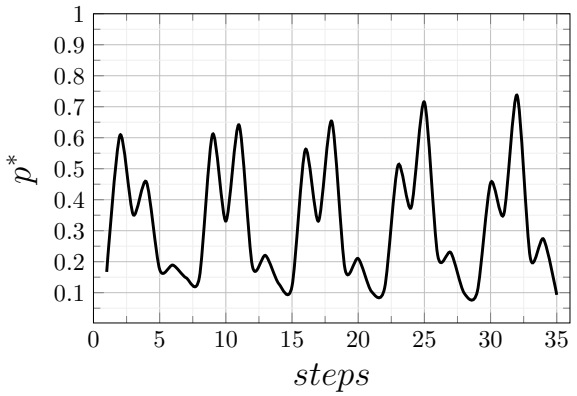

Figure 7: Probability profile of measuring the marked node with Grover coin

Thanks to the simulation of this exact circuit with the addendum script of [7], out intuition suggests that our implementation complies with the structure theorised by Thomas Wong.

5. Quantum Information Set Decoding Algorithm

Employing the same quantum exploration approach, we have implemented a hybrid classical-quantum version of the FS-ISD. The first step is to encode V_{up1} and V_{up2} in the quantum circuit and to obtain an efficient way of selecting their columns. This is accomplished by applying a series of **CNOT** on two registers $|sel\rangle$ and $|sum\rangle$. The first is used as selector of the columns and the second to compute the sum between the activated columns. An example is shown in Fig. 8.

$$V_{up1}=\begin{bmatrix}1&1&0&0\\1&0&1&1\\1&1&1&0\\0&1&0&1\end{bmatrix} V_{up2}=\begin{bmatrix}1&0&1&0\\0&1&1&0\\1&1&0&1\\1&1&1&1\end{bmatrix} \hat{s}_{up}=\begin{bmatrix}1\\1\\0\\0\end{bmatrix}$$

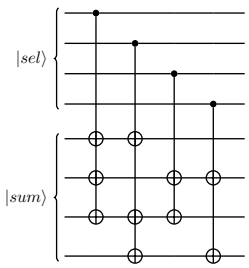

Figure 8: The column selectors circuit for V_{up1}

By using the binary encoding of the Johnson graph in Fig. 4 for the selectors, we are able to express all the possible $\binom{(k+l)/2}{p/2}$ way of combining the columns of the two matrices. Following the work from Kachigar and Tillich [3], to find a collision for Equation 1 we employ a Quantum walk search on the cartesian product of two Johnson graphs. With the values for V_{up1},V_{up2} and $\hat{\mathbf{s}}_{up}$ reported above the new search space can be expressed as $J^2(4,2) = J_1(4,2) \times J_2(4,2)$. A graphic representation is shown in Fig. 9.

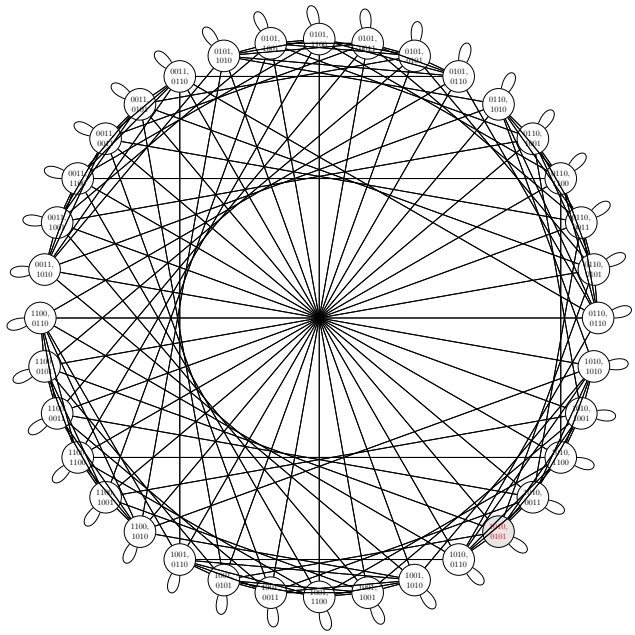

Figure 9: $J^2(4,2) = J_1(4,2) \times J_2(4,2)$ with marked vertex {1010,0101} highlighted in red

Finding the marked vertex {1010,0101} in the $J^2(4,2)$ graph corresponds to the selection of the columns in position 0 and 2 of V_{up1} and in position 1 and 3 of V_{up2}, since they sum to the right values of $\hat{\mathbf{s}}_{up}$.

$$\begin{bmatrix}1\\1\\1\\0\end{bmatrix}\oplus\begin{bmatrix}0\\1\\1\\0\end{bmatrix}\oplus\begin{bmatrix}0\\1\\1\\1\end{bmatrix}\oplus\begin{bmatrix}0\\0\\1\\1\end{bmatrix}=\begin{bmatrix}1\\1\\0\\0\end{bmatrix}=\hat{s}_{up} \tag{6}$$

The self-loop weight value is now

$$\gamma = 2^{\lceil \log_2 \binom{(k+l)/2}{2} \rceil} - \frac{p}{2}\left(\frac{k+l}{2} - \frac{p}{2}\right) \tag{7}$$

The Quantum collision algorithm requires four register:

- $|v_1\rangle$ is the state register for $J_1((k+l)/2, p/2)$, every qubit corresponds to a position in the set $\frac{k+l}{2}$ thus it has dimension $dim(|v_1\rangle) = (k+l)/2$

- $|v_2\rangle$ is the state register for $J_2((k+l)/2, p/2)$, every qubit corresponds to a position in the set $\frac{k+l}{2}$ thus it has a dimension $dim(|v_1\rangle) = (k+l)/2$
- $|sum\rangle$ is the ancilla register: it encodes the matrix values and its dimension depends on the number of rows of V_{up} so $dim(|sum\rangle) = l$
- $|c\rangle$ is the coin register: it encodes all the possible direction of the coin $dim(|c\rangle) = \lceil \log_2 \binom{(k+l)/2}{2} \rceil$

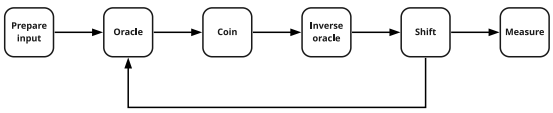

Figure 10: High level overview of our algorithm

The algorithm can be summarized in six stages, as shown in Fig. 10:

- **Input preparation**: in order to generate the superposition of the nodes of the two Johnson graphs $J_1((k+l)/2, p/2)$ and $J_2((k+l)/2, p/2)$ we apply the Dicke operator on registers $|v_1\rangle$ and $|v_2\rangle$. On the other hand, to obtain an equal superposition of all the direction we apply the Hadamard gate on the coin register $|c\rangle$.
- **Oracle**: we encode the values of V_{up1} and V_{up2} on $|sum\rangle$. The selectors for the first matrix are activated by $|v_1\rangle$ and for the second by $|v_2\rangle$. At the end of the oracle phase we obtain on register $|sum\rangle$ the syndrome value corresponding to the sum of the selected columns.
- **Coin**: the Grover diffuser operator is applied on register $|c\rangle$ regardless of the state value and in order to mark the right vertex the application of the second coin is conditioned by the desired value of the syndrome.
- **Inverse oracle**: this phase is used to uncompute the register $|sum\rangle$ to reset the register for the next walk iteration. This is performed by applying in reversed order the same sequence of operations of the oracle phase.
- **Shift**: it has the same structure of Fig. 6 but is applied twice to the two state registers simultaneously in both graphs during the same application.
- **Measurement**: after repeating the stages in the loop body $t^* = \frac{\pi}{\sqrt{2(\gamma+1)}}\sqrt{\binom{(k+l)/2}{p/2}}$ times we measure registers $|v_1\rangle$ and $|v_2\rangle$.

The circuit representation is reported in Fig. 11. The final circuit's performance regarding gate number and depth is described in Tables 1 and 2.

6. Conclusions

The general goal of this thesis has been to explore the usefulness of Quantum Random Walks in ISD algorithms. Current literature on the topic suggests studying Johnson graphs because of their regularity. This led us to find a circuital implementation of such walks on a Johnson graph, which to the best of our knowledge has never been accomplished.

Taking inspiration from the work of Kachigar and Tillich, which conceptualized the use of Quantum random walks to solve ISD problems, we set up to develop a quantum algorithm that finds collisions on a sub-procedure of the Finiasz-Sendrier ISD algorithm.

We succeded in obtaining a quantum circuit that performs this operation. We simulated the circuit using Qiskit, IBM's quantum circuit simulation library, and compared it with Thomas Wong's mathematical formalization of Lackadaisical Quantum Random Walks obtaining strong evidence that our solution complies with this type of walk.

Our solution could be employed in any search-based problem involving a structure which can be encoded as a Johnson graph or a product of them.

The most challenging part was figuring out an efficient way of performing the shift operation in a Johnson graph which turned out to be quite complicated due to the combinatorial nature of its encoding. The initial approach of having a loop-free structure has been discarded in favour of a significant reduction in computational resources. The final solution has linear spatial complexity but the maximum probability of measuring the correct element is not close to 1, as Thomas Wong's work explains, and requires a relatively high number of iterations because of the exponential depth of the Shift phase.

Regarding future development, different paths are possible. Firstly, research should focus on

	Input Preparation	Oracle	Coin	Inverse Oracle	Shift
Gates	$\mathcal{O}(\frac{k+l}{2}+p^2)$	$\mathcal{O}(\frac{k+l}{2}l)$	$\mathcal{O}(\log\binom{(k+l)/2}{2})$	$\mathcal{O}(\frac{k+l}{2}l)$	$\mathcal{O}(\binom{(k+l)/2}{2}\log\binom{(k+l)/2}{2})$
Depth	$\mathcal{O}(\frac{k+l}{2})$	$\mathcal{O}(\frac{k+l}{2})$	10	$\mathcal{O}(\frac{k+l}{2})$	$\binom{(k+l)/2}{2}$

Table 1: $J^2((k+l)/2, p/2)$ performance

	Input Preparation	Oracle	Coin	Inverse Oracle	Shift
CNOT	$5(k+l)p/4 - 5p^2/4 - (k+l)$	$(k+l)l$	0	$(k+l)l$	0
RY	$(k+l)p - p^2 - 2(k+l) + 1$	0	0	0	0
H	$\lceil\log_2\binom{(k+l)/2}{2}\rceil$	0	$4\lceil\log_2\binom{(k+l)/2}{2}\rceil$	0	0
X	0	0	$4\lceil\log_2\binom{(k+l)/2}{2}\rceil$	0	0
CZ	0	0	2	0	0
CSWAP	0	0	0	0	$2\binom{(k+l)/2}{2}\lceil\log_2\binom{(k+l)/2}{2}\rceil$

Table 2: $J^2((k+l)/2, p/2)$ gate numbers

understanding if different ISD algorithms can benefit from this quantum solution. Secondly, finding a more efficient solution for the shift phase in terms of both depth and gate number could drastically improve the circuit performance. Our intuition suggests that a first step in this direction could involve finding an indexing algorithm for combinatorial encodings as shown in [4]. It would be interesting to understand whether the same problem can be solved using other graph topologies on which an efficient shift can be implemented. This work focuses on the specific model of the coined quantum walk. Nevertheless, we do not exclude the possibility of obtaining benefits from using other search paradigms such as the Szegedy Quantum walk model or the Continuous time Quantum Walk.

7. Acknowledgements

We want to thank prof. Barenghi and prof. Pelosi for introducing us to this innovative topic as well as continuous guidance and support and Simone Perriello for the laughs as well as key tips and tricks he gave us during the last 12 month. We also thank ATOS for providing us access to their quantum simulators.
Finally, we thank our families, friends and cats

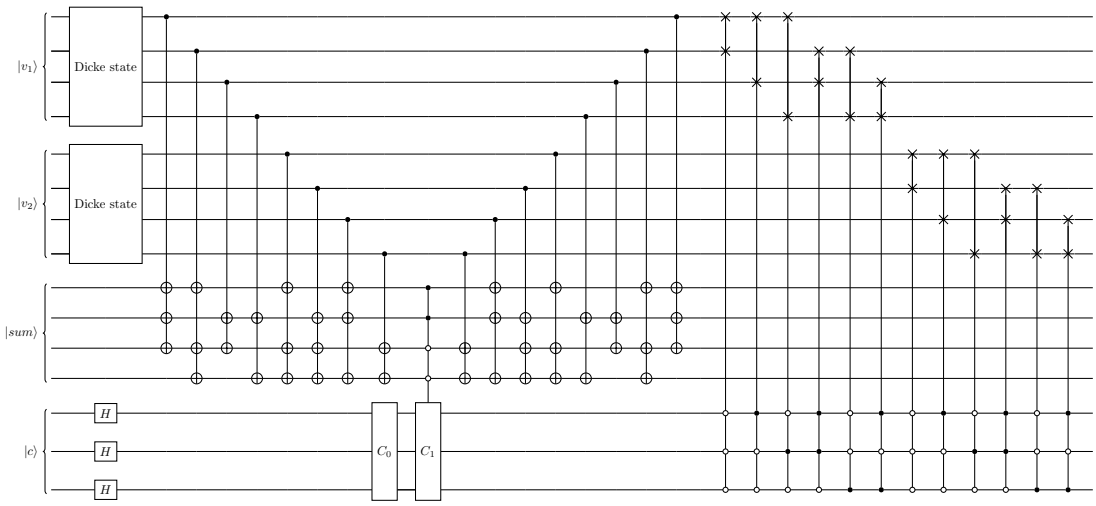

Figure 11: Example of one step of a Quantum random walk search on a product of $J(4,2) \times J(4,2)$ with V_{up1}, V_{up2} and $\hat{\mathbf{s}}_{up}$ values described in Equation (1)

for supporting us during this adventure.

References

[1] Y. Aharonov, L. Davidovich, and N. Zagury. Quantum random walks. *Phys. Rev. A*, 48:1687–1690, Aug 1993.

[2] Matthieu Finiasz and Nicolas Sendrier. Security bounds for the design of code-based cryptosystems. In Mitsuru Matsui, editor, *Advances in Cryptology - ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings*, volume 5912 of *Lecture Notes in Computer Science*, pages 88–105. Springer, 2009.

[3] Ghazal Kachigar and Jean-Pierre Tillich. Quantum information set decoding algorithms. In Tanja Lange and Tsuyoshi Takagi, editors, *Post-Quantum Cryptography - 8th International Workshop, PQCrypto 2017, Utrecht, The Netherlands, June 26-28, 2017, Proceedings*, volume 10346 of *Lecture Notes in Computer Science*, pages 69–89. Springer, 2017.

[4] S. Marsh and J. B. Wang. Combinatorial optimization via highly efficient quantum walks. *Physical Review Research*, 2(2), jun 2020.

[5] Chandra Sekhar Mukherjee, Subhamoy Maitra, Vineet Gaurav, and Dibyendu Roy. Preparing dicke states on a quantum computer. *IEEE Transactions on Quantum Engineering*, 1:1–17, 2020.

[6] Neil Shenvi, Julia Kempe, and K. Whaley. A quantum random walk search algorithm. *Physical Review A*, 67, 10 2002.

[7] Thomas G. Wong. Faster search by lackadaisical quantum walk. *Quantum Inf. Process.*, 17(3):68, 2018.