QUANTUM ORACLES - HOW TO TRANSFORM CLASSICAL PROBLEMS INTO QUANTUM ONES

Alexandre Silva
Computer Science
UNIVEM - Centro Universitário Eurípides de Marília

Luis Hilário Tobler Garcia
Computer Science
UNIVEM - Centro Universitário Eurípides de Marília

Maúricio Duarte
Information Technology
Fatec Garça – Deputado Julio Julinho Marcondes de Moura

August 23, 2024

ABSTRACT

Using quantum oracles and other effects, like superposition, *5 mini-projects* were done. The main goal of these projects was to answer if it's possible to bring some problems to the quantum realm and if such translation worth it. After finishing the tests, it was possible to see that, some of the implementations show descent results. However, classical computing is still a fundamental part of quantum algorithms.

1 Introduction

Now days, isn't difficult to hear someone talking about quantum computers, and how these machines will change the world. Nonetheless, the major fraction of these comments come from extrapolations and science fiction present in popular movies and series around the world. In this paper, I'm going to show that quantum computing it's not a magical trick to solve everything, but a tool for solving a distinct group of problems.
To do that, *5 mini-projects* were implemented using qiskit, an open source framework from IBM, exploiting quantum effects and using classical/quantum algorithms to reach the expected outcomes. After that, the results will be here compared with their classical counterparts. Such mini-projects are the following: Quantum File explorer3.1, miles to kilometers conversion3.2, Hanoi towers3.3, Buckshot Roulette 3.4 and QRAM 3.5. All these implementations can be found at my GitHub repository.

2 Oracles

Based on the idea of *Oracle Turing Machines* [1][2][3][4], the Oracles are mathematical modeling tools, used to abstract outlying parts of an algorithms into a black-box, making the algorithm analysis much easier. These machines could also be seen as a function, getting an input x e returning $f(x)$ with time-complexity $O(1)$. Because of these unreachable characteristics for real life, this model can't be implemented, being used only for formal description of decision problems.
However, in quantum computing, it's possible to implement something similar to that, taking advantage of your inner structure and quantum effects to gain a *Speed-up* relative to its classical counterparts, such Speed-up can be seen, for example, in the Deutsch–Jozsa algorithm [5]. Furthermore, the Quantum Oracles have a fundamental role determining the circuit complexity. Some approaches for that are: *depth*, calculating the longest path in the circuit that some information must pass through and *gate counting*, summing up how many gates were applied in the final circuit. Nevertheless, these approaches are very dependent of the *QPU* topology, differing from each *backend* used during the *transpilation* process. To solve this problem, a well-known technique is to pack some parts of the circuit into Oracles,

and then describing the complexity based on how many times they are called, it's also known as *query complexity* [6] [4].

2.1 Types of Oracles

Using the base description of Quantum Oracles, we can classify them based on their structures and how the data is processed.

2.1.1 Phase Oracle

The Phase Oracle, is the most well-known format. Some Algorithms, like Deutsch–Jozsa, Grover, Simon and Bernstein–Vazirani, use it to take advantage over Classical approaches.

2.1.1.1 Default Behavior

As its main characteristic, the Phase Oracle adds a global phase to the circuit, using quantum effects like *Phase Kickback* (basically the phase pass all the way through CNOT's target and is applied in the control Qubit), to change values in superposition.

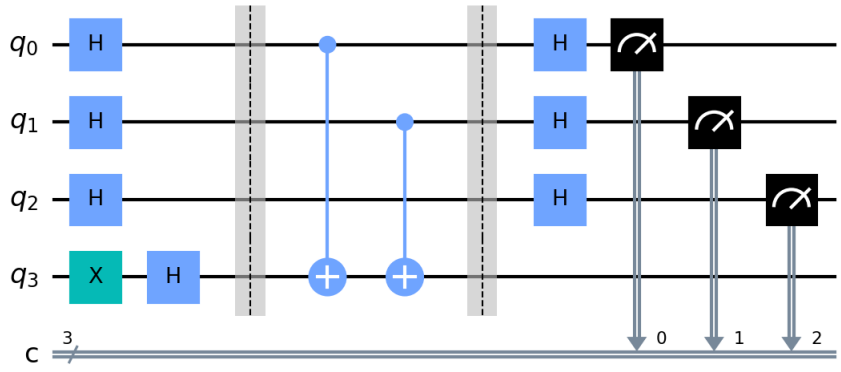
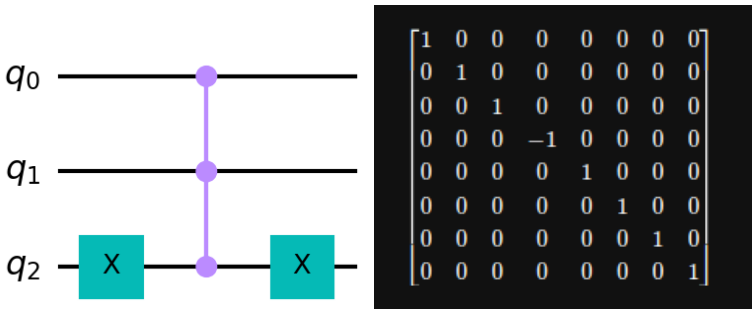


Figure 1: Phase Oracle using Phase-Kickback Example

In the Figure 1, a π phase was added in the auxiliary Qubit ($q3$), taking advantage of the phase $|-\rangle$, which will be the responsible to modify the values in the unitary matrix. In this setup, the $CNOT$ gates act slightly different from its default behavior, this way, the gate invert the value of the target Qubit and, due the π phase, it also acts like a Z gate applied in the control Qubit. So, after applying $CNOT\,|-\rangle\,|+\rangle$ (little-endian), the state becomes $\frac{1}{\sqrt{2}}(|0\rangle\,|-\rangle - |1\rangle\,|-\rangle)$, and after removing the superposition, the final state is: $\frac{1}{\sqrt{2}}(|+\rangle\,|1\rangle - |-\rangle\,|1\rangle)$. This way, the control Qubit is changed due the *Phase Kickback* and its state is flipped from $|0\rangle$ to $|1\rangle$. Taking this effect as and advantage, we can use it to encode some binary values inside these oracles and use them to do some calculations.

2.1.1.2 Minimal Oracle Version

Furthermore, There's another layout possible for implementing a Phase Oracle. Once this one only applies a phase in some bit-strings, the auxiliary Qubit can be removed, and the phase can be provided by CZ gates (or any other gate which can apply a phase for certain bit-strings). Doing that, it's possible to create a phase oracle keep its unitary and reversible nature, in the format of a minimal Oracle.



(a) Phase Oracle - Minimal Oracle Layout Example

(b) Figure 2a Resulting Unitary Matrix

Figure 2: Phase Oracle Minimal Version

In the example above 2, a MCP gate was applied adding a π phase and another two X gates were used to invert the Qubits we want to have the value 0 as a trigger for the MCP phase application. Following the little-endian format the final encoded bit-string is: 011_2 or 3_{10}.

Looking at its unitary matrix 2b, it's possible to see the 8×8 identity matrix with a -1 value in the column relative to the bit-string 011_2, representing the encoded value.

2.1.2 Boolean Oracle

The Boolean Oracle has a similar structure with the Phase Oracle. However, this time a phase isn't applied, acting this way a regular boolean function, mapping inputs to outputs.

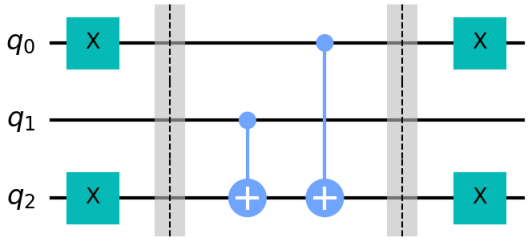


Figure 3: Boolean Oracle Example

The example in the Figure 3, is a well-known balanced Oracle for the Deutsch-Jozsa's algorithm. Nevertheless, to implement it, the Oracle must be transformed into a Phase Oracle.

2.1.3 Minimal Oracle

As previously mentioned, the Minimal Oracle is a function which its essence is unitary and reversible, so no additional Qubits are required. Therefore, this format can be either Boolean or Phase Oracle, depending on its implementation.

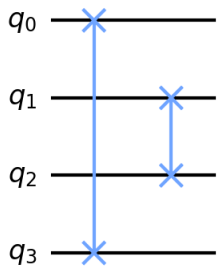


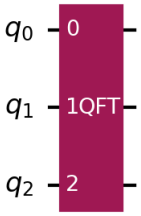
Figure 4: Boolean Minimal Oracle example

The example 4 shows an Boolean Oracle which two $SWAP$ gates were used to invert the Qubits values order. Doing this way, the final matrix keeps its unitary and reversible properties, once the $SWAP$ gates act in both Qubits together, mirroring its action.

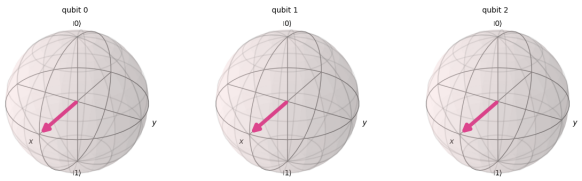
2.1.4 QFT(Quantum Fourier Transform)

The QFT is, in a nutshell, a quantum algorithm based on the Discrete Fourier Transform, used for quantum states period finding, projecting values from the computational basis onto the X basis (also know as Fourier basis).

Even though it's an algorithm by itself, its application is done in the format of Oracles in quantum circuits.



(a) QFT Oracle example



(b) Values Mapped onto Fourier basis example (quantum state $|000\rangle$)

Figure 5: QFT Oracle

2.1.5 Other kinds of Oracles

In addition to the oracles mentioned above, it's possible come across with different mentions of Oracles, like the Simons's, Deutsch-Jozsa's, etc. However, these are just implementations of models already shown in this paper. Also, the most relevant for the following projects are the Phase and Boolean ones, due that, there's no use to keep discussing about many others sub-categories of Oracles here.

3 Development

3.1 File Explorer

Imagine a quantum computer, but instead of one of those we already have in real life, imagine a different kind, very similar to those we have at home, with a quantum operational system, quantum files, etc. This imaginary version could be thought as a hybrid computer as well, taking advantage of both quantum and classical capabilities. Using this idea, the first project implements a file explorer for this ideal system.

3.1.1 Algorithms Used

3.1.1.1 Grover

The Standard algorithm for search problems is the Grover's algorithm. This one, realizes searches on unsorted "databases" (bit-string in this case) with $O(\sqrt{2^n})$, being n the number of Qubits. Its tenets are based on amplitude amplification, using a Phase Oracle to mark some bit-strings and then using a Diffuser to amplify the probability to find these values after measuring it.

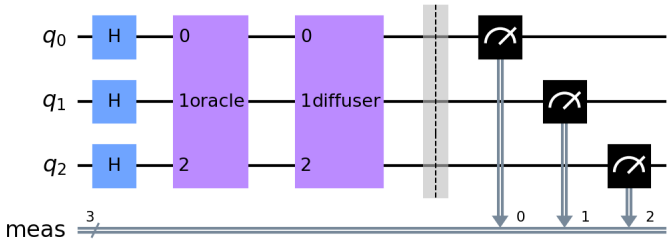


Figure 6: Example Grover's algorithm with 3 Qubits

For building the circuit, is needed a joint of $Oracle + Diffuser$ applied k time, which $k \approx \frac{\pi}{4\sqrt{\frac{a}{2^n}}} - \frac{1}{2}$, being a the number of values marked by the Oracle. Once we want to find just a single file, the use of this relation is not required, being need to apply the algorithm once to find the bit-string.

Even this being the best option for bit-string finding, in this project different rotations were tested trying to find a better superposition which increases the chances to find the value we want.

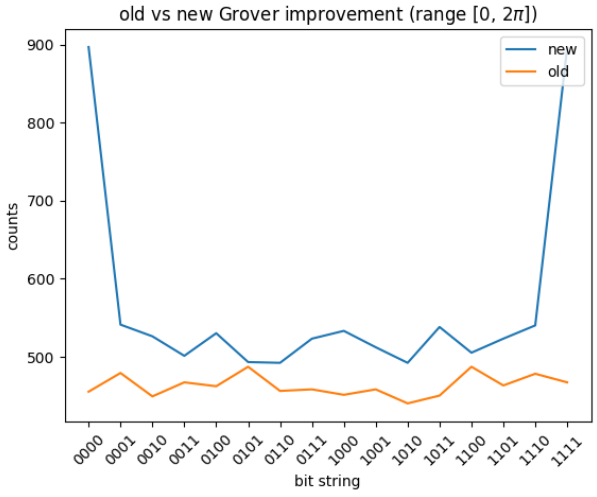


Figure 7: Comparing the Standard Grover's algorithm with the version modifying the rotation angles $[0, 2\pi]$ for each 4 bits bit-string

Using specific rotations for each bit-string, it's possible to improve the outcomes, and even stand out the default rotation.

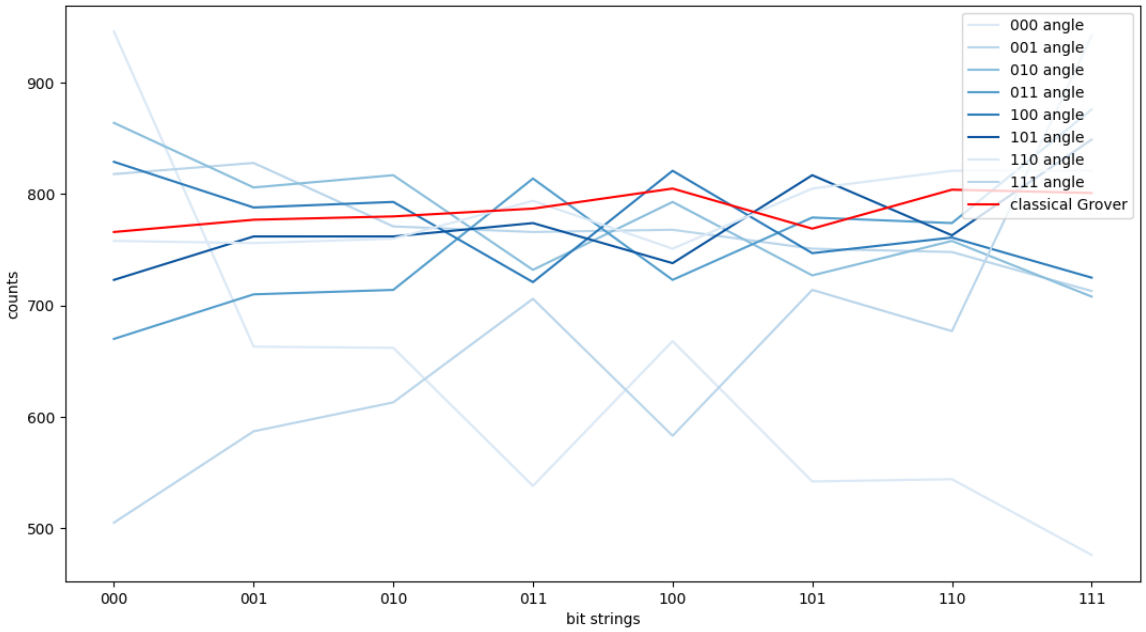
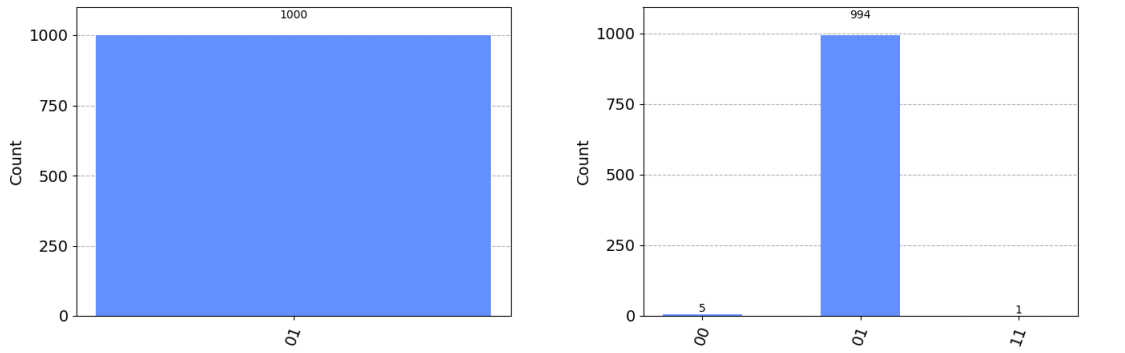


Figure 8: Test Using the best angles of each bit-string in different ones

However, using the best angle of a distinct bit-string as the source of superposition in the algorithm, isn't enough to overcome the classical Grover implementation, and also the probability distribution becomes irregular. So, for a general purpose implementation, the default rotation is the best one for the most part of the cases. This behavior is even weird with 2 Qubits, being the H gate rotation the best ever for that.



(a) Results for Grover with 2 Qubits (default rotation)

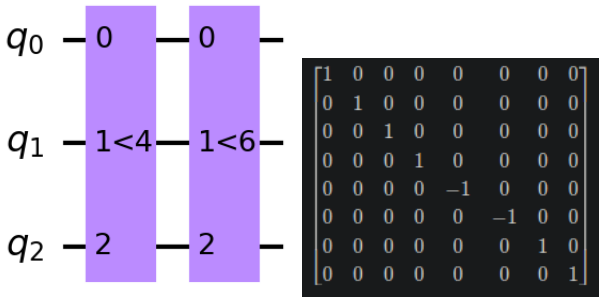
(b) Results for Grover with 2 Qubits (modified rotation)

Figure 9: Comparing Grover with 2 qubits using the default and modified rotation

For our purpose in this project, the hybrid approach was used. So during the circuit setup, the searched value pass through a Hash-Table will optimal angles. This way, the algorithm can achieve better outcomes at the end of the execution, finding, this way, the file we want.

3.1.1.2 Sets Difference

Overlapping two distinct Phase Oracles, each one with a range of values encoded. The remaining Phases, are the difference of the sets encoded by them [7].



(a) Sets Difference Algorithm Example

(b) Unitary After applying the Sets Difference algorithm

Figure 10: Sets Difference example

In the last example 10a, the set $\{000, 001, 010, 0110\}$ was encoded in the first Oracle and $\{000, 001, 010, 011, 100, 101\}$ in the second one.

After Overlapping the phases 10b, only the values $\{100, 101\}$ keep the -1 value. This way, we can do the set operation just applying some oracles with the values range we want.

3.1.2 Final Solution

As a solution for this problem, a hash function was made, mapping the path of some file p to a specific bit-string b, $F : p \to b$. With this function, we can use a set returned values and encode them into a Phase Oracle, creating a sort of quantum Look-Up-Table for files in that machine. Moreover, a second Oracle containing all the bit-strings, with the exception of the searched one, is applied for difference of sets algorithm, this way the phases will be overlapped, and only the one we want will pass to the Grover's algorithm.

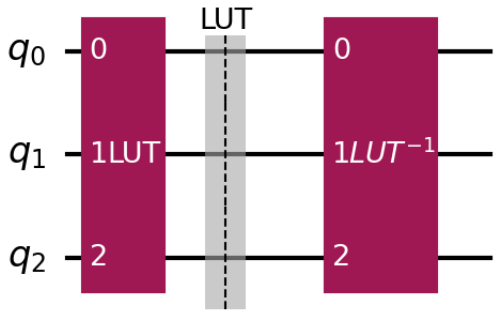


Figure 11: Look-up-Tables applied

Finally, the modified version of the Grover's algorithm is applied.

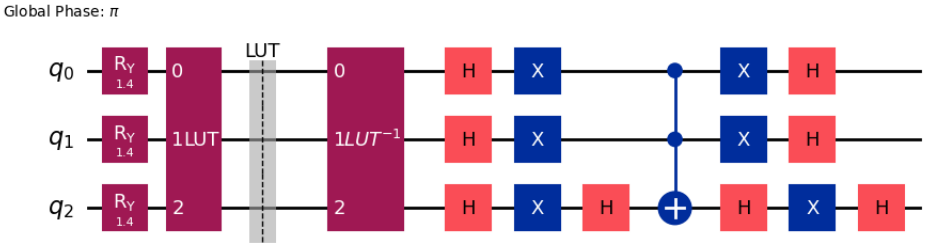


Figure 12: File explorer final circuit

This setup, maximizes the probability to find the queried file, presenting the following distribution after 1000 shots.

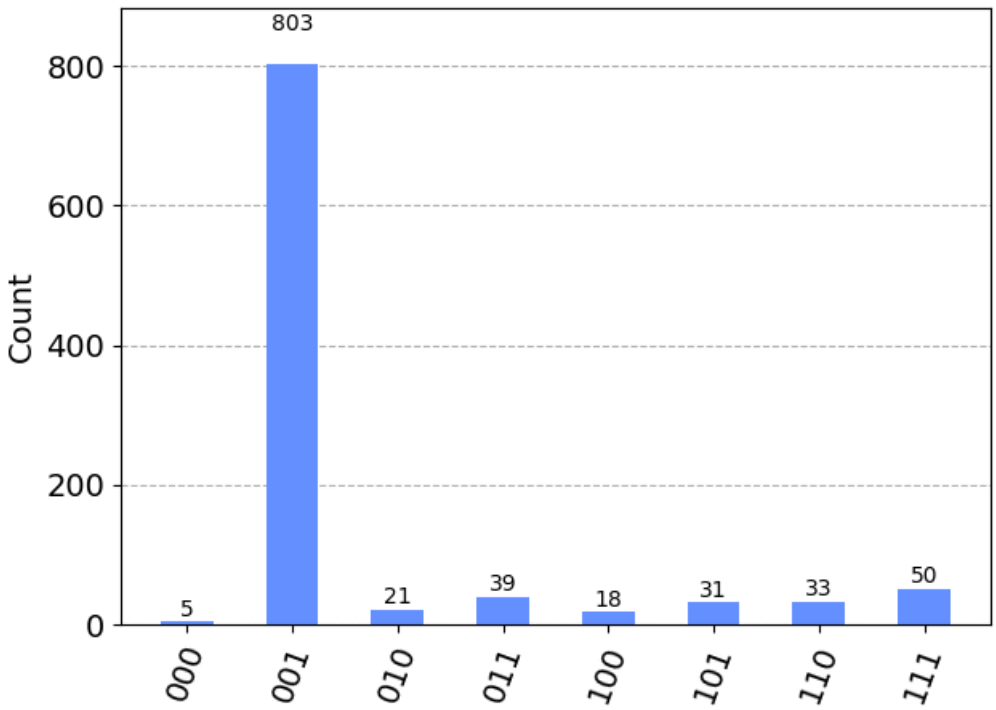


Figure 13: File Explorer Results after running with Qiskit-AER

3.1.3 Results

For this specific ideal case, it is, probably, one of the best known ways to do the search between "quantum files".

However, it's not the best alternative as a way to speed-up classical searches. It happens because storing a large Look-Up-Table for encoded files and another one for keeping track of each optimal angle for every bit-string contained between 2^n combinations, could be really costly and slow. Also, it would be required a hash function with a tiny probability of collision.

An alternative for that could be using the standard Grover's algorithm, but it still the need to encode the hard drive files to a big table. Also, comparing the classical tree approach with Grover's, their worst cases scenarios are $O(log(n))$ and $O(\sqrt{n})$, which implies that the first one still faster than the later. As a last point, the quantum version for this problem, could present the wrong file after searching due errors and the quantum randomness.

Keeping that in mind, the final algorithm is a right choice for a quantum system as describe at the beginning of this section, but it can't overcome the best algorithm in a classical computer.

3.2 Miles to Kilometers Conversion

The second problem that was tested here, was the conversion from miles to kilometers. This idea came out after finding a quantum algorithm capable of calculate the Fibonacci sequence, which is an essential piece for this very project.

3.2.1 Used Algorithms

3.2.1.1 Quantum Fibonacci

The quantum version of Fibonacci was presented in [8]. This paper shows that, using a quantum circuit with all bit-string in superposition, and then apply controlled rotations to remove values that contains consecutive ones, is possible to approximate the result of the *n*-th Fibonacci sequence value.

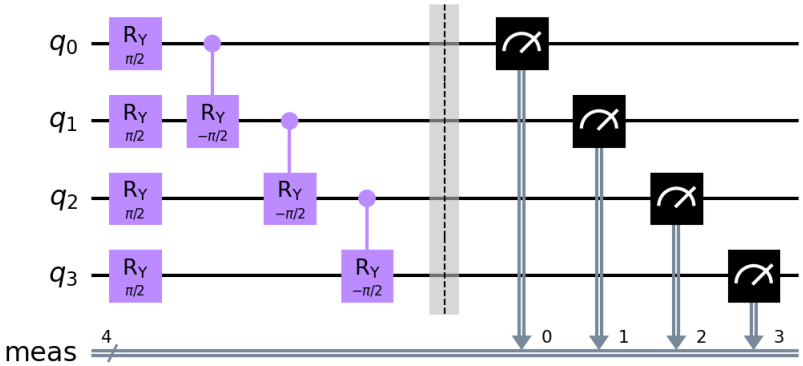


Figure 14: Quantum Fibonacci Example

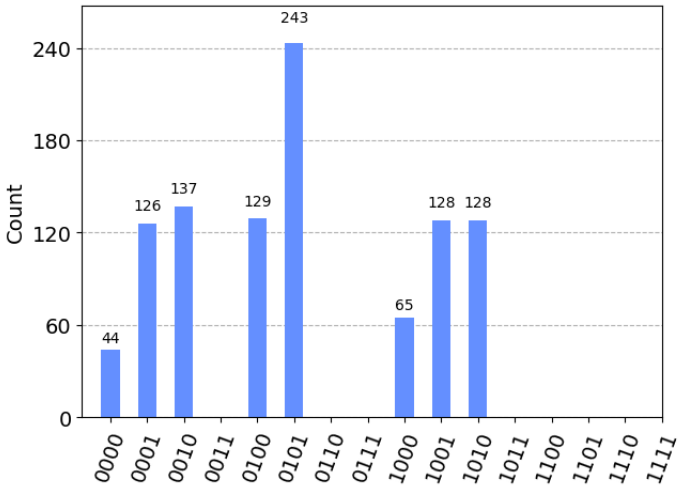


Figure 15: Quantum Fibonacci result - F(4)

After running the circuit k times, the amount of unique bit-string that appeared during the experiment represent the n-th Fibonacci value, which n is the total of Qubits were used in the circuit. So, for the example 14 above, the position n we want, is 4, and the result after counting the bars of which values are $\neq 0$ give us the result of $F(4) = 8$.

It may seem wrong, however, the quantum setup differ slightly from the classical one, once the first value is 2 instead of 1, this way, the sequence is: $F(1) = 2, F(2) = 3, F(3) = 5, F(4) = 8, F(5) = 13, F(6) = 21, ...$, which implies that it worked for this example.

3.2.1.2 Approximating Miles to Kilometers

To approximate the value from miles to kilometers, an old approach for that is using the Fibonacci sequence. This method, uses the relation $F_{km} = F_{milhas}(n + 1)$, which F is the classical Fibonacci algorithm ($F(1) = 1$ and $F(2) = 2$), and n is a known position in this very sequence. Following this equation, the relative kilometers values will be given in the next n $(n + 1)$.

miles	Km	Real(Km) value
1	2	1.60934
2	3	3.21869
3	5	4.82803
5	8	8.04672

Table 1: Approximated miles-Km values

For values that aren't present in the sequence, they can be generated combining multiple values which are there. For example, to transform 10 miles to kilometers, we could use the values 8 and 2, which are known in the sequence, and then used the previous cited method for each part. So, for this example, $8 = F(5)$ and $2 = F(2)$, so $F(5+1) + F(2+1) = F(6) + F(3) = 13 + 3 = 16Km$ giving a close value of $\approx 16.0934Km$.

3.2.2 Implementation

Using the formulation above, the final algorithm takes place using a Hybrid approach, following the structure bellow:

Algorithm 1 Quantum Conversion algorithm

parts = $BreakInputIntoFibonacciParts(input)$
for parte in parts **do**
    Apply the Oracle embedding $F(part)$
    Measure the Oracle Qubits
    Reset these Qubits
end for
Check the outcomes
Multiply the given result for its relative i

In this algorithm, is needed a preprocessing routine, using a classical algorithm, to split the input number into smaller parts that can be calculated using the sequence. Also, to reduce the amount of Oracle applications to decrease the quantum noise, this processing returns a tuple for each number containing n and the total of applications needed p, this way, we can apply each Oracle once and after retrieving the outcomes, the final result is giving by: $\sum_{i=0}^{m} o_i p_i$, being m the total of parts and o the given output.

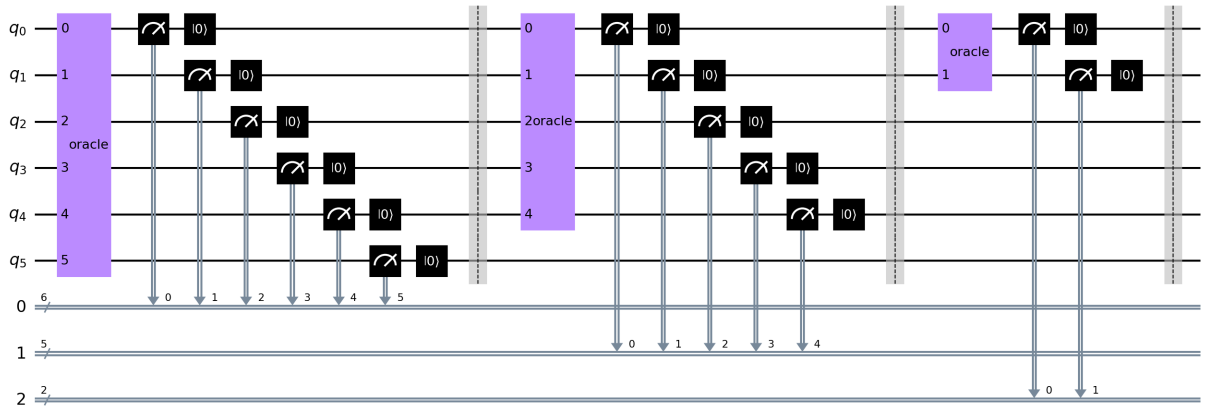


Figure 16: Final Miles to Kilometers Circuit

3.2.3 Thoughts on the Results

With this method, is possible to evaluate some input values. However, there are some points which make it unfeasible.

Execution Time Each part to be calculated, takes too much *shots*, taking around $500 - 10000$ to get better results, therefore increasing the execution time

Errors Due the high depth of this kind of circuit, this setup is likely to present wrong results due multi-Qubit operations in real hardware

Impression After running the algorithm for many different values, it's clear that, for large numbers, the output deviates a lot from what the expected was

Classical Intervention Once it requires both pre and post-processing using classical algorithms, the usefulness of the quantum version decreases

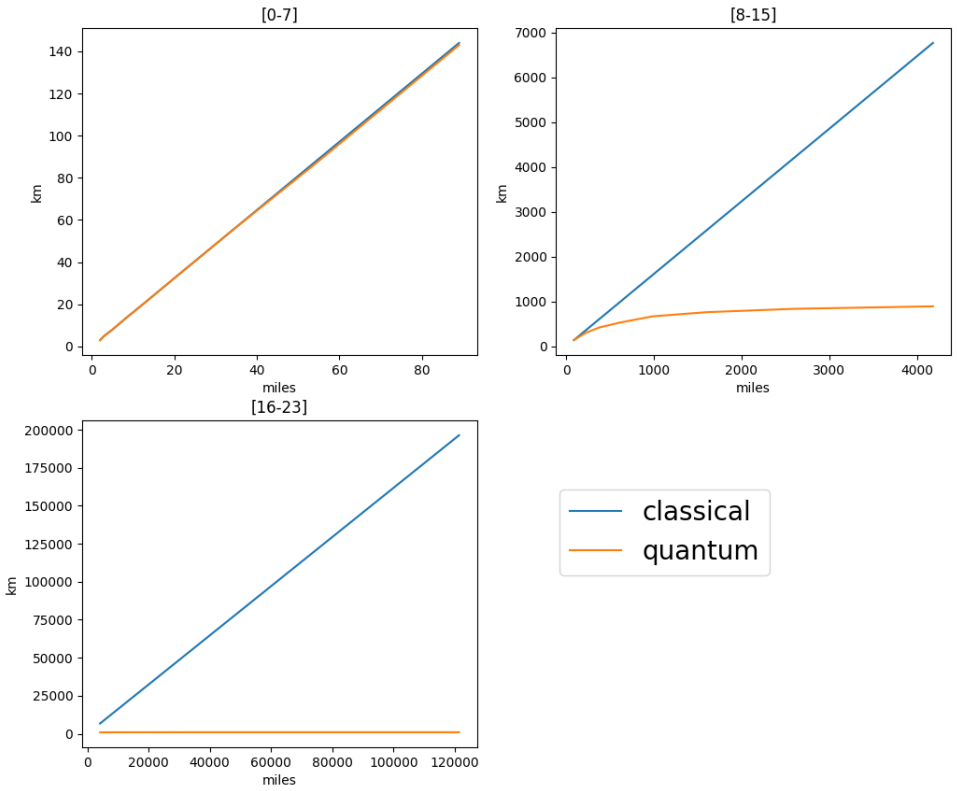


Figure 17: Comparing the precision between classical and quantum versions

Thus, this protocol can be used for small numbers. However, to be useful as its classical version, it should be more resilient to errors and precision issues, also it needs to be more independent from classical computations, once most of the work is based on classical routines.

Consequently, this final algorithm isn't good as the classical one, and probably isn't useful enough for real applications. Nonetheless, it's possibly a starting point for different approaches for real implementations.

3.3 Hanoi Towers

For the Hanoi Towers implementation, it was designed a way to encode the discs positions in the tower using their binary values and the Phase Oracle as a storage device.

3.3.1 Implementation

This project, requires $(\lfloor \log_2 x \rfloor + 1) * 3$ Qubits, being x the number of disks. The Qubits layout follows this sequence: $|t_{n-1}t_{n-2}...t_0\rangle\,|a_{n-1}a_{n-2}...a_0\rangle\,|s_{n-1}s_{n-2}...s_0\rangle$, where s,a,t are the first, second and third towers respectively.

To prepare the state, a Phase Oracle is prepared encoding each disc number (from 1 to x) in the first rod (Qubits s) using the global phase π. After that, $SWAP$ operations are done following the classically preprocessed steps, moving bit-by-bit from one rod to another.

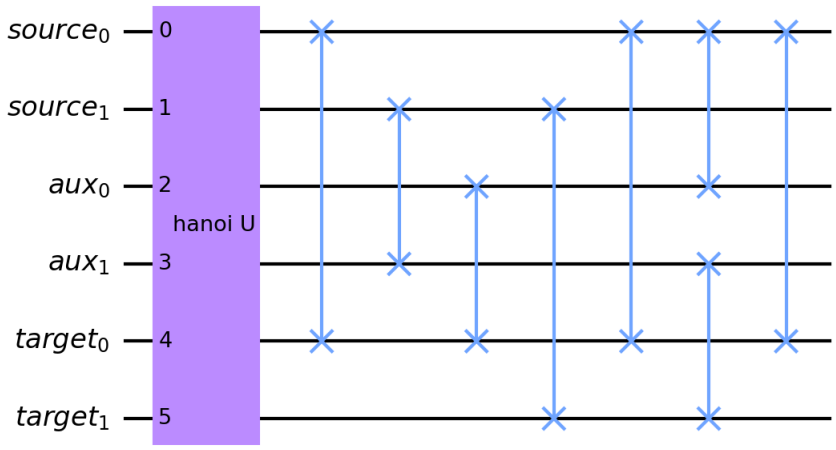


Figure 18: Quantum Hanoi Tower with 3 disks

As cited previously, the movements are preprocessed before playing. This way,the quantum setup acts like a classical player, executing step-by-step his strategy.

Also, as we're playing with phases, it's possible to apply phase amplification algorithm, like Grover's, to evaluate the tower and get the resulting values.

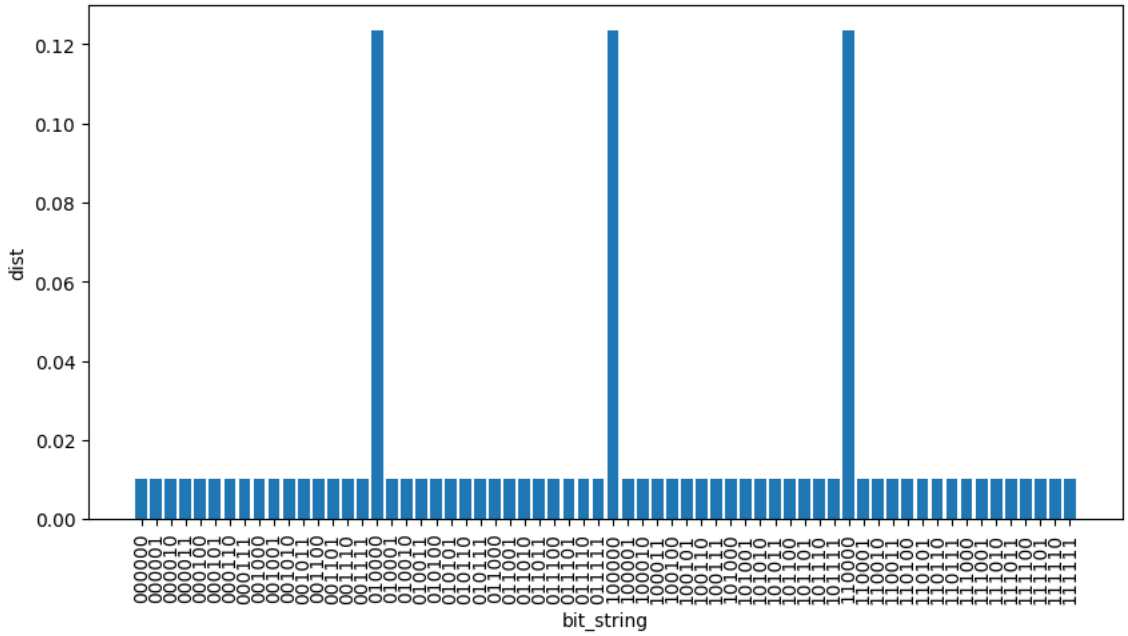


Figure 19: Quantum Hanoi Tower with 3 disks - evaluation using Grover's algorithm

As shown in 19, after evaluating the circuit, it's possible to see 3 high probability bit-strings, which represent the encoded disks. These bit-string are 010000, 100000 and 110000, so the operations moved successfully the disks from the source rod (rightmost 2 bits) to the target rod (leftmost 2 bits).

3.3.2 Thoughts on the Results

Once it followed the same sequence of implementation as the classical one, requiring, inclusively, preprocessing. This version is equivalent to the classical algorithm, keeping in mind that for a classical computer, it's also required to precompute a movement, and then play it.

3.4 Buckshot Roulette

Buckshot Roulette is a computer game made by Mike Klubnika, based on the idea of modifying the infamous Russian Roulette.

During the game, you are challenged to play against a demon (dealer). If you win, you'll receive a prize, otherwise the game will restart and you can try again.

In this very project, we analyzed the first game match, trying to find the best strategy to maximize the player's winnings. The reason for choosing the first match, is because of its simplicity, being away from power-ups that could complicate the analysis, and also because this one shows the base dynamics for the entire game, so doing well in this one, is the first step to beat the whole game.

3.4.1 Game Dynamics

In the first round, 2 fake bullets and 1 real are put into the shotgun. Then, the player starts his play, choosing between shooting himself or his opponent. If the player shoots himself and it's a real bullet, he loses one life point, otherwise he gets another chance to choose. However, if he chooses to shot his opponent, if it's real he takes away one life point from him, otherwise his opponent receives the gun.

This dynamics is followed in the entire game for both player and dealer, only being slight modified when power-ups are gained.

3.4.2 Modeling the Game Strategies

The round dynamics can be modeled as a binary tree, using some game theory ideas to understand it better.

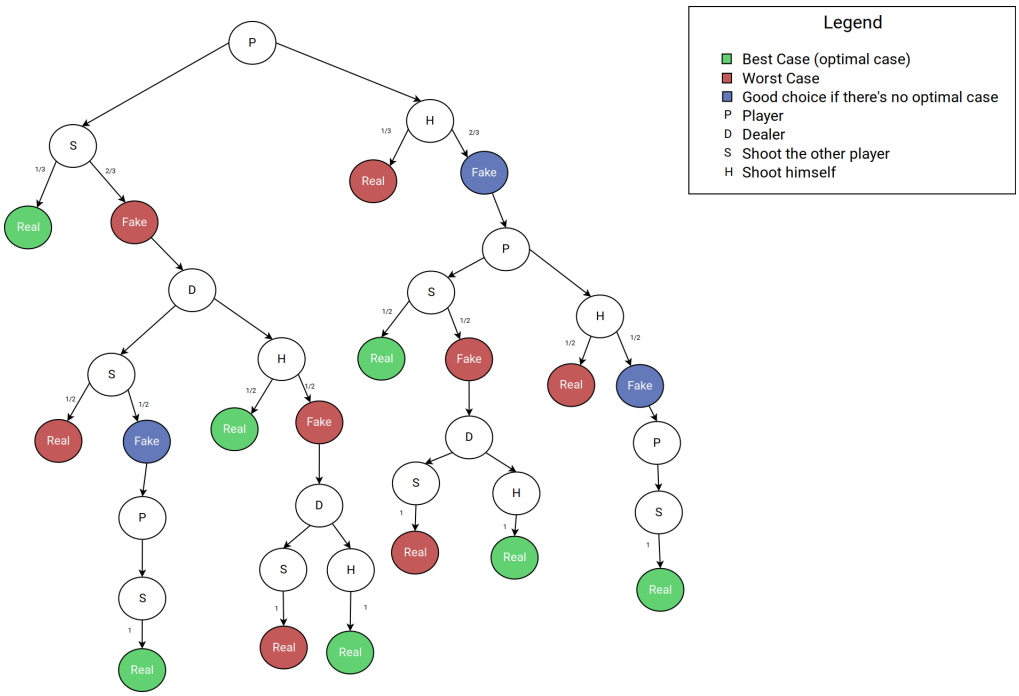


Figure 20: Buckshot Roulette tree diagram

For this diagram, it's expected that the player is a rational agent, which always plays aiming to win, and the dealer is a random agent, choosing randomly its decisions. Due that, it's possible to see in the image 20 that the leafs that was reached by the player, has only one possible value, Shoot, but those reached by the dealer, has two, Shoot himself or his opponent, once he's a random agent.

Using this tree, we can create an algorithm to simulate this game multiple times, and then try to find a possible solution.

After testing multiple possible paths, the best one was given by the path which the player starts shooting the dealer. After Analyzing this very strategy, the conclusion is that, this is the best strategy, because the player has less chances to die, once he doesn't have the chance of killing himself right away at the beginning.

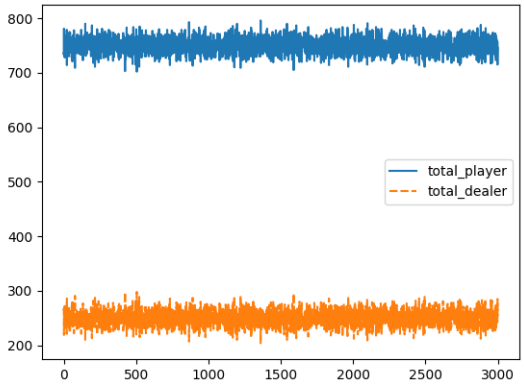


Figure 21: Classical Buckshot Roulette - Best Strategy

Bullet	Action	action result	shoot result
1	player shoots himself	real	dealer wins
1	player shoots himself	fake	-
2	dealer shoots player	real	dealer wins
2	dealer shoots player	fake	-
2	dealer shoots himself	real	player wins
2	dealer shoots himself	fake	-
3	player shoots dealer	real	player wins
3	dealer shoots player	real	dealer wins
3	dealer shoots himself	real	player wins

Table 2: Player starts shooting himself path analysis

Bullet	Action	action result	shoot result
1	player shoots dealer	real	player wins
1	player shoots dealer	fake	-
2	dealer shoots player	real	dealer wins
2	dealer shoots player	fake	-
2	dealer shoots himself	real	player wins
2	dealer shoots himself	fake	-
3	player shoots dealer	real	player wins
3	dealer shoots player	real	dealer wins
3	dealer shoots himself	real	player wins

Table 3: Best Strategy - Analyzing the possible paths

Analyzing these tables, it's possible to see that the path of which the player starts shooting his opponent 3, he has one chance more to win than when he shoots himself first 2. Due this fact, the 3 shows the best Strategy to beat the dealer.

3.4.3 Quantum Version

After analyzing the game, a quantum circuit was modeled to mimic the game dynamics. To do that, an Oracle was implemented for each player (player and dealer) internally implementing their strategies based on Bloch Sphere rotations.

Also, a *Shoot* gate was implemented. This one, is a controlled Hadamard gate, which implies that the gun has 50% chance of firing (bit 1) and 50% of not firing (bit 0), as well.

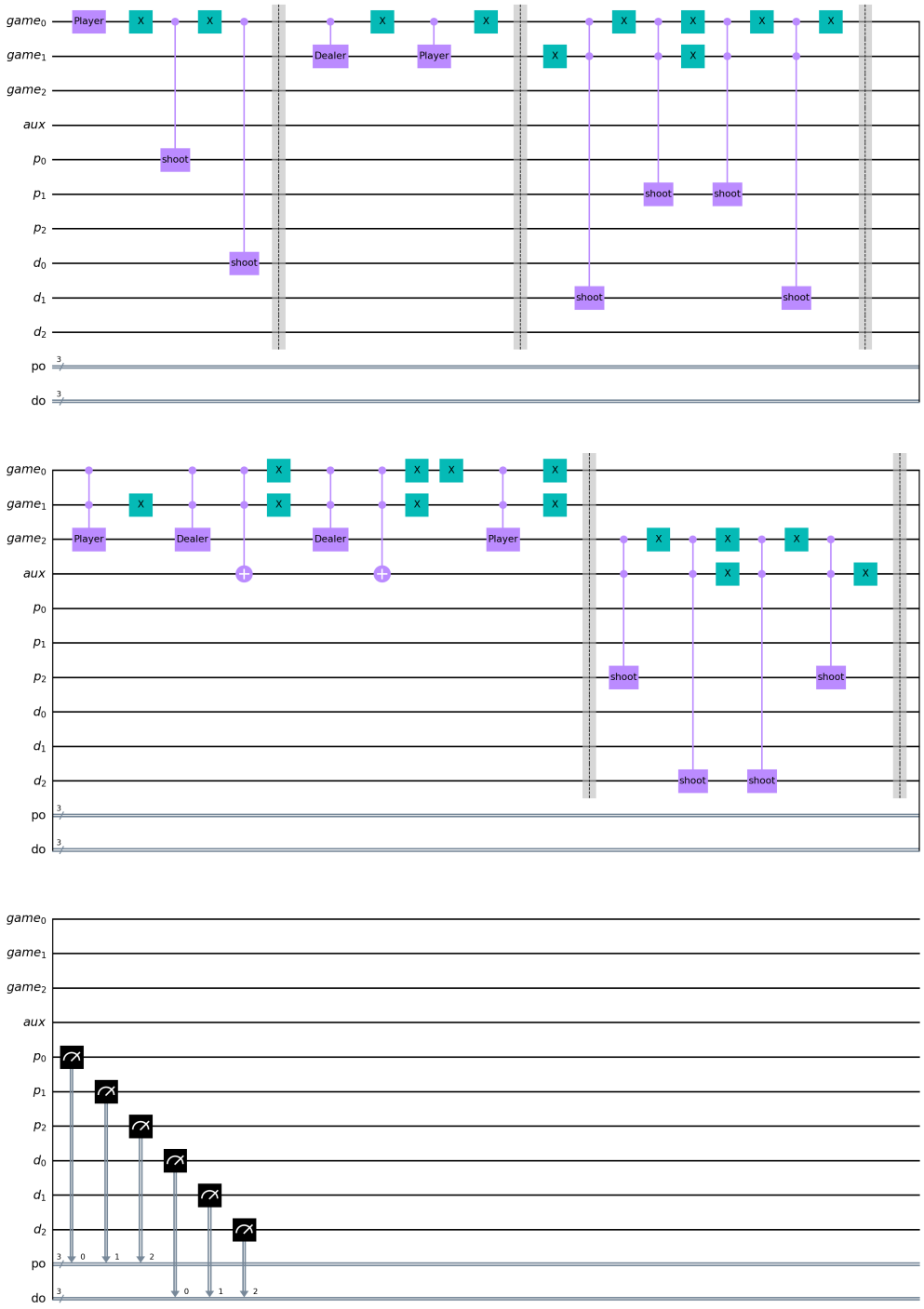
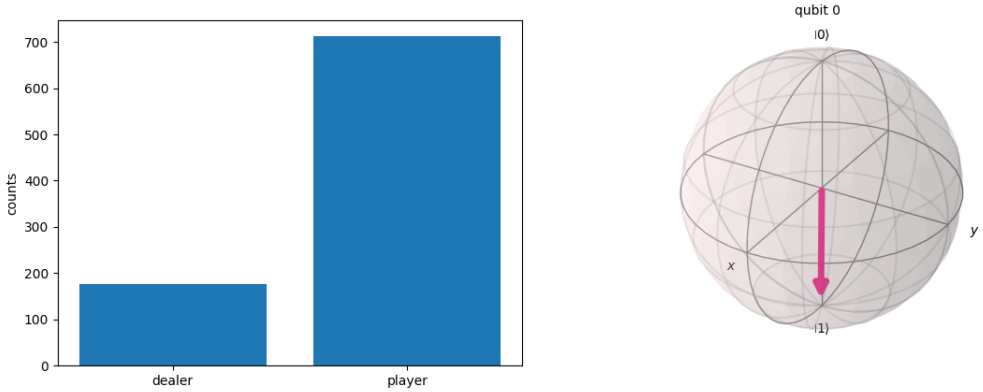


Figure 22: Buckshot Roulette Circuit

After setting-up the circuit, two parameters were inserted into Player's Oracle. These were used to manipulated the rotations in the Bloch Sphere in relation to both θ and ϕ angles. This way, it was possible to simulate it multiple times and find the best angles which raise the maximum values of player winnings. After running the algorithm, the best values were: $\theta \approx 3.0853981633974477, \phi \approx 3.7853981633974474$ radians.



(a) Buckshot Roulette (Qiskit Aer) simulation ($shots = 1000$) results

(b) Best Strategy Bloch Sphere plot

Figure 23: Buckshot Roulette results

Plotting the simulation results on a histogram 23a, it's possible to see that the total sum doesn't reach the total of 1000 shots which were done. It happened due the layout of the circuit, which doesn't take cares of a "game over" state, and keeps running even if the bullet was already shot. Due that, it needed a post-processing function to handle these cases.

Aside of this detail, the results kept really solid, and, comparing to the classical version, the results seem to be the same, and even the player strategy keeps the same, being the best choice shooting his opponent instead of himself 23b (in this case, 1 represents the action of shooting the opponent for both players).

3.4.4 Conclusions

For this problem, there are no competition between classical and quantum versions, once the former is highly inspired in the later. However, taking a more game theoretical view, the quantum version can take advantage of more states, consequently it can explore more strategies than its counterpart.

In general, both simulations reached the same result, and it was shown that it's possible to model and simulate a game with quantum circuits using Oracles as player's strategies representations.

3.5 QRAM

Finally, the last project done was a *QRAM* based on Quantum Oracles. In this version was tested a *QROM*, with static data inside, and a test for a QRAM able to write data inside.

Differing from most part of researches on these devices, in this research the goal was to store all kinds of quantum states, not only bit-strings. The reason for this choice is the real application for this project, if someday a real QRAM was built, its maximum power would be based on storing superposition quantum states.

3.5.1 QROM

To implement a QROM, n Qubits are used for the address bus and m for the data bus, but these numbers are not correlated, so it's possible to have a configuration like $n = 3; m = 10$ using the sequence $|0\rangle^{\otimes m}|a_{n-1}a_{n-2}...a_0\rangle$. This happens due the address mapping circuit. In the circuit, when a bit-string address is put on its bus, controlled gates are used to handle the input and map this to the desired state, so the whole application can be done with more than one gate and more than one target Qubit.

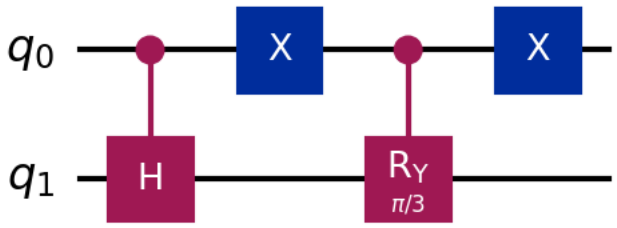


Figure 24: QROM circuit example

In this example 24, q_0 represents the address bus and q_1 the data bus. With this setup, two states were mapped for both possible addresses. In this case, when the value present in the address 0, the circuit will trigger the controlled $RY(\frac{\pi}{3})$ gate, applying it on q_1. The same happens for 1, but this time, a controlled H is applied, putting the target Qubit in half superposition.

Taking this whole circuit, it can be put inside a Quantum Oracle, and then use it onto a bigger circuit, even taking advantage of superposition for all states with the QROM, just adding either all or some addresses in superposition.

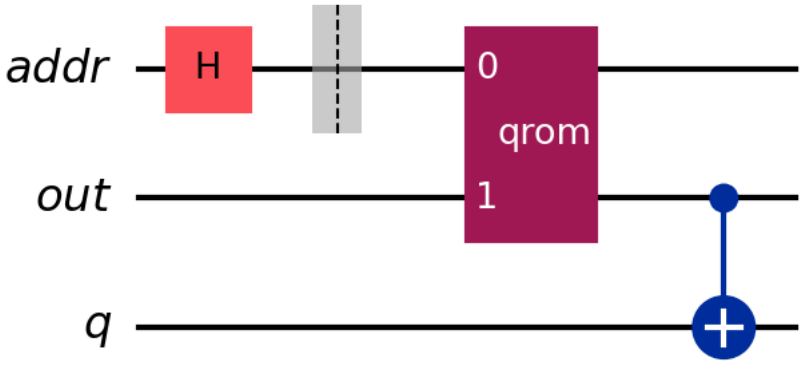


Figure 25: Applying superposition on a QROM

Here 25, all possible addresses were put on superposition, and then the resulting states are entangled with q .

Although it seems kinda simple, due the no-cloning-theorem, it's not possible to copy internal values and add it in different Qubits. This way, we can only use the state as a entanglement source, or use quantum teleportation to transfer the state from a Qubit to another one.

Also, this kind of memory can become really large depending on its goal, and is required to be reapplied each time it's requested.

3.5.2 QRAM

For the QRAM, the quantum teleportation was used, as cite before. With it, we can have a circuit with n qubits, being each Qubit a single address, and then use teleportation to move a state from QRAM's domain to the external circuit domain.

In this example case, the first n *data* Qubits act both as data as address, however for teleport their states, additional n Qubits (in this case the Qubits labeled as *t*) being in total $2 * n$ Qubits required for the whole protocol.

With that, our circuit Qubits q are prepared with some arbitrary states we want to save, then the teleportation protocol is applied and the previous stored data is removed from the circuit and moved to the QRAM domain.

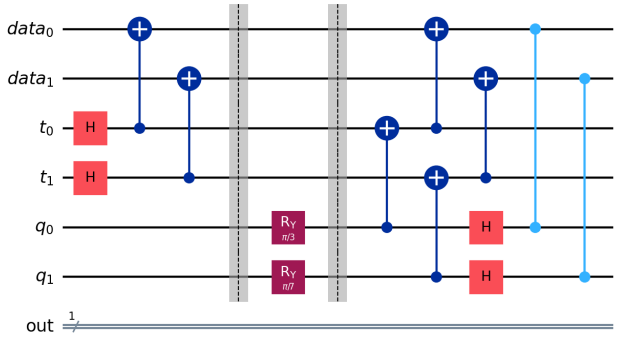


Figure 26: QRAM circuit example

Comparing this implementation with previous one, we have similar characteristics, once both QRAM and QROM can't have their values copied. However, this one doesn't require reapplication for writing, and retrieving values, once they were pre-applied at each data Qubit. Also, it's possible to overwrite and interfere over data estates, this way it's possible to reduce the memory size just managing stored values.

3.5.3 Conclusion

Although it's possible to create some sort of memorization inside a circuit, as retreaded at [13] and [14], these devices are far away from a real application. Even using the circuits shown above, the real usage is not simple and practical, being needed many applications, clever data managing protocols and many Qubits, for implementing. It's also affected several implications that reduce its usefulness, like: no-cloning, decoherence, errors, etc.

Due that, the best approach could be implementing memory on hardware, using peripheral devices to connect it to the software and them save or retrieve data when needed.

Based on this discussion, it's possible to conclude that, although it's possible to create sort of memory circuits for small use cases, the extrapolation for bigger ones is far from real.

3.6 Final Thoughts

After all this work, it's evident that Quantum Computing has a big potential for the future. However, the lack of hardware capabilities and some quantum effects affect the real use cases for quantum computing now.

As already shown by many researches in areas like: chemistry, physics, cryptograph, optimization, etc. Quantum computing can be the next breakthrough in technology, and a decisive point for many companies around the world.

However, during NISQ era, to increase the usefulness of such technology, it's still required using classical computing along with it, for both pre- and post-processing routines. This way, using Hybrid approach, it's possible to get the best from both, exploiting the their finest algorithms.

In conclusion, it's possible to take advantage of quantum computing now for some problems we know classically. But, it's needed to investigate if there are some quantum factors that can improve of worsen the results. In case you find out something that can be beneficial for your goal, sometimes taking a hybrid approach on the problem can be a better choice instead of tackling it with pure quantum computations.

References

[1] Robert I. Soare. Turing oracle machines, online computing, and three displacements in computability theory. *Annals of Pure and Applied Logic*, 160(3):368–399, 2009. Computation and Logic in the Real World: CiE 2007.

[2] Sadika Amreen and Reazul Hoque. Oracle turing machines.

[3] Subrahmanyam Kalyanasyndaram. mod04lec23 - oracle turing machines, 09 2021.

[4] Niklas Johansson and Jan-Åke Larsson. Quantum simulation logic, oracles, and the quantum advantage. *Entropy*, 21(8), 2019.

[5] Yale Fan. A generalization of the deutsch-jozsa algorithm to multi-valued quantum logic. In *37th International Symposium on Multiple-Valued Logic (ISMVL'07)*. IEEE, May 2007.

[6] Ryan O'Donnell. Lecture 5: Quantum query complexity, 09 2015.

[7] Javier Sanchez-Rivero, Daniel Talaván, Jose Garcia-Alonso, Antonio Ruiz-Cortés, and Juan Manuel Murillo. Some initial guidelines for building reusable quantum oracles, 2023.

[8] Austin Gilliam, Marco Pistoia, and Constantin Gonciulea. Canonical construction of quantum oracles, 2020.

[9] Lídia André. Tower of hanoi – lídia andré, 03 2021.

[10] diptokarmakar47. How to solve the tower of hanoi problem - an illustrated algorithm guide, 01 2019.

[11] Towers of hanoi: A complete recursive visualization, 05 2020.

[12] GeeksforGeeks. Program for tower of hanoi, 05 2014.

[13] Samuel Jaques and Arthur G. Rattew. Qram: A survey and critique, 2023.

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

[15] Dave Bacon. Cse 599d -quantum computing simon's algorithm, 2006.

[16] Robin Kothari. An optimal quantum algorithm for the oracle identification problem. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2014.

[17] Ryan O'Donnell. Lecture 13: Lower bounds using the adversary method, 10 2015.

[18] Laurel Brodkorb and Rachel Epstein. The entscheidungsproblem and alan turing, 12 2019.

[19] Martin Davis. Turing reducibility?, 11 2006.

[20] Mahesh Viswanathan. Reductions 1.1 introduction reductions, 2013.

[21] Takashi Yamakawa and Mark Zhandry. Classical vs quantum random oracles. Cryptology ePrint Archive, Paper 2020/1270, 2020. `https://eprint.iacr.org/2020/1270`.

[22] Harry Buhrman, Richard Cleve, and Avi Wigderson. Quantum vs. classical communication and computation, 1998.

[23] Elham Kashefi, Adrian Kent, Vlatko Vedral, and Konrad Banaszek. Comparison of quantum oracles. *Physical Review A*, 65(5), May 2002.

[24] William Zeng and Jamie Vicary. Abstract structure of unitary oracles for quantum algorithms. *Electronic Proceedings in Theoretical Computer Science*, 172:270–284, December 2014.

[25] Alp Atici. Comparative computational strength of quantum oracles, 2004.

[26] Kathiresan Sundarappan. How to build oracles for quantum algorithms, 04 2022.

[27] Zhifei Dai, Robin Choudhury, Jinming Gao, Andrei Iagaru, Alexander V Kabanov, Twan Lammers, and Richard J. Price. View of the role of quantum algorithms in the solution of important problems.

[28] Don Ross. Game Theory. In Edward N. Zalta and Uri Nodelman, editors, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Spring 2024 edition, 2024.

[29] Tomasz Zawadzki and Piotr Kotara. A python tool for symbolic analysis of quantum games in ewl protocol with ibm q integration. `https://github.com/tomekzaw/ewl`.

[30] Piotr Frackiewicz. Application of the ewl protocol to decision problems with imperfect recall, 2011.

[31] Jens Eisert, Martin Wilkens, and Maciej Lewenstein. Quantum games and quantum strategies. *Physical Review Letters*, 83(15):3077–3080, October 1999.

[32] Muhammad Usman. Kilometres to miles conversion — approximation of fibonacci series, 09 2019.

[33] Faisal Shah Khan and Ning Bao. Quantum prisoner's dilemma and high frequency trading on the quantum cloud. *Frontiers in Artificial Intelligence*, 4, 11 2021.

[34] Alexis R. Legón and Ernesto Medina. Dilemma breaking in quantum games by joint probabilities approach. *Scientific Reports*, 12, 08 2022.

[35] Brian Siegelwax. Quantum memory: Qram. what is it and why do we need it? making quantum algorithms thrive., 01 2022.

[36] Gabriel Landi. Density matrices and composite systems.

[37] V. Vijayakrishnan and S. Balakrishnan. Role of two-qubit entangling operators in the modified eisert–wilkens–lewenstein approach of quantization. *Quantum Information Processing*, 18, 03 2019.

[38] Real Python. Scientific python: Using scipy for optimization – real python.

[39] scipy optimize minimize scalar scipy $v1.12.0$ manual.

[40] Matt Davis. Optimization (scipy.optimize) — scipy v0.19.0 reference guide.

[41] scipy.optimize.minimize — scipy v1.6.0 reference guide.