
QUANTUM ORACLES - COMO TRANSFORMAR PROBLEMAS CLÁSSICOS EM QUÂNTICOS

✉ Alexandre Silva

Ciências da Computação

UNIVEM - Centro Universitário Eurípides de Marília

ABSTRACT

1 Introdução

Hoje, não é difícil ver alguém falando sobre computação quântica e como essas máquinas vão mudar o futuro. Contudo, muitas dessas frases acabam se levando por extrapolações e/ou usos indevidos de ficção. Neste artigo, mostrarei que nem tudo é possível ser feito com um computador quântico atual, assim como existem pequenas áreas que se beneficiam ao máximo dessa nova tecnologia.

Para esse feito, serão mostrado alguns testes feitos usando o qiskit, um framework open source da IBM para computação quântica, além de alguns resultados obtidos após executar os algoritmos em simuladores e máquinas reais, assim como seus relativos em computação clássica. Algoritmos dos quais tomam proveito dos quantum oracles, modelos ideias de função que não ajudam a descrever o algoritmo matematicamente, também tomam proveito de alguns efeitos quânticos, como superposição e interferência, para se sobressair à algumas estratégias clássicas.

Com isso, o projeto foi desenvolvido em cima de cinco pequenos problemas, sendo eles: conversão de milhas para quilômetros, torre de Hanoi, explorador de arquivos, Buckshot Roulette e QRAM. Todas as implementações e materiais utilizados podem ser encontrados nesse repositório do GitHub.

2 Início do projeto

Para dar início a pesquisa, foi necessário entender quais os tipos de oracles existem e como eles podem ser usados.

Em computação clássica, temos as Oracle Machines, as quais são máquinas de Turing, das quais implementam alguma função em seu interior, e ao ser chamado/invocado o resultado correto é retornado em tempo constante $O(1)$, podendo ser vista como uma caixa preta, abstraindo completamente o seu funcionamento. Devido a essa definição, as OM's são ideias matemáticas, sendo assim usados apenas para formalismo matemático.

Contudo em computação quântica, podemos de fato implementar certos modelos de Oracles e adiciona-los a um circuito maior, executando certas funções como: encoding de dados, aplicação de $f(x)$, abstração de partes do circuito, etc.

2.1 Tipos de Oracles

2.1.1 Phase Oracle

Um dos primeiros tipos de Oracles usados para a criação de algoritmos como os de: Grover e Deutsch-Jozsa; é comumente conhecido como *Phase Oracle*.

Tal dispositivo, é usado para atribuir uma fase ao circuito, sendo muito usado para configurar valores, explorar a interferência ou se aproveitar de outros efeitos como o *Phase Kickback*. Matematicamente poderíamos descrever ele da seguinte forma: $|x\rangle|-\rangle \rightarrow (-1)^{f(x)}|x\rangle|-\rangle$, do qual $|x\rangle$ é a entrada do oracle e $|-\rangle$ é a ancilla que prove a fase.

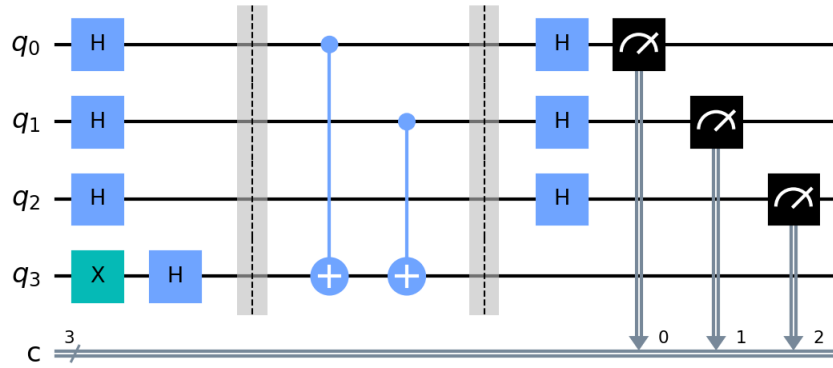


Figura 1: Exemplo de phase oracle usado para o algoritmo de Deutsch–Jozsa

No exemplo acima, utilizamos o *Phase Kickback* para adicionar uma fase nos qubits 0 e 1, transformando seus estados de $|+\rangle$ para $|-\rangle$, fazendo com que ao serem colapsados o resultado $|1\rangle$ apareça na saída.

É possível também criar um phase oracle removendo o qubit adicional (nesse exemplo o Q3), uma vez que podemos utilizar outros gates para introduzir a fase e manter ainda natureza unitária.

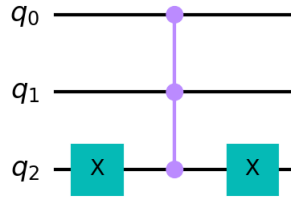


Figura 2: Exemplo fase Oracle sem a Ancilla

Dessa vez, utilizamos o *MCP* gate para adicionar uma fase global π e dois gates *X* para dizer quais qubits queremos q tenham o valor 0, codificando assim o valor 011 ou 3 na base decimal.

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

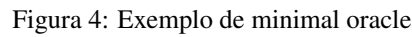
Figura 3: Matriz unitária do Phase oracle

É possível verificar então que ao criarmos esse circuito, matemos a matriz identidade e adicionamos a fase -1 no valor da coluna relativa ao 011.

Essa versão pode ser considerada como um minimal oracle, uma vez que a própria função interna se mantém unitária, sem a necessidade de ancilla.

O Boolean oracle, por sua vez, representa uma função booleana, sem qualquer adição de fases. Nesse caso, $|x\rangle$ representa a entrada do oracle e $|y\rangle$ representam os qubits auxiliares que receberam a resposta, $|x\rangle |y\rangle \rightarrow |x\rangle |y \oplus f(x)\rangle$.

Como já citado anteriormente, o minimal oracle possui uma função que em sua essência é unitária, não requerendo qubits adicionais $|x\rangle \rightarrow |f(x)\rangle$.



2.4 Simon's Oracle

Nesse algoritmo, configuramos uma chave s dentro do oracle, e ao executar o algoritmo temos os possíveis períodos da função, sendo necessário rotinas de pós processamento para identificar o valor correto.

Por fim, o Oracle QFT aplica a versão quântica da transformada de Fourier, projetando os valores de entrada na base X (também conhecido como base de Fourier).

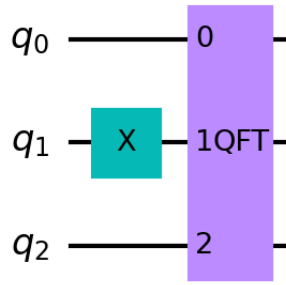


Figura 6: Exemplo do algoritmo de QFT

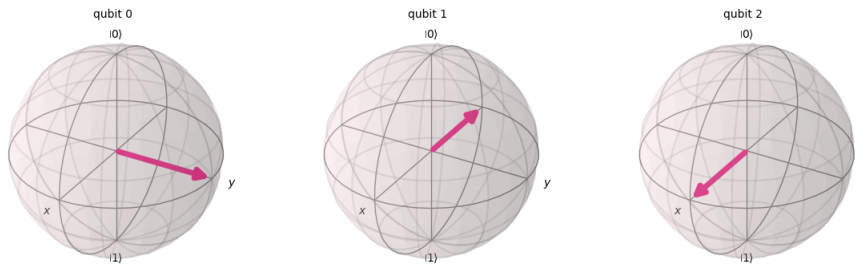


Figura 7: Valores mapeados na base de Fourier

3 Desenvolvimento

3.1 File Explorer

Imagine um computador quântico com um sistema operacional quântico (semelhante aos computadores convencionais, mas dessa vez seguindo as leis da mecânica quântica). Pensando nas partes desse sistema operacional, como seria possível pegar arquivos da memória usando a computação quântica?

3.1.1 Algoritmos usados

3.1.2 Grover

Um dos algoritmos mais comuns para a área é o algoritmo de Grover. Esse algoritmo realiza buscas em "bancos de dados" (bit strings) desorganizados em tempo $O(\sqrt{2^n})$ onde n é o número de qubits usados. Nele, usamos um circuito do qual amplifica-se a probabilidade de encontrar os valores marcados no Oracle na saída. Tal algoritmo segue o seguinte padrão:

1. Configuramos todas as possíveis bit strings, ou seja aplicamos uma superposição uniforme $H^{\otimes n}$
2. aplicamos o phase oracle U_f do qual implementa uma função que marca os valores que desejamos encontrar
3. aplicamos o operador de Grover Diffuser, $\mathbb{I} - 2|s\rangle\langle s|$, sendo s o estado com o valor que queremos

Convertendo para um circuito temos algo semelhante a:

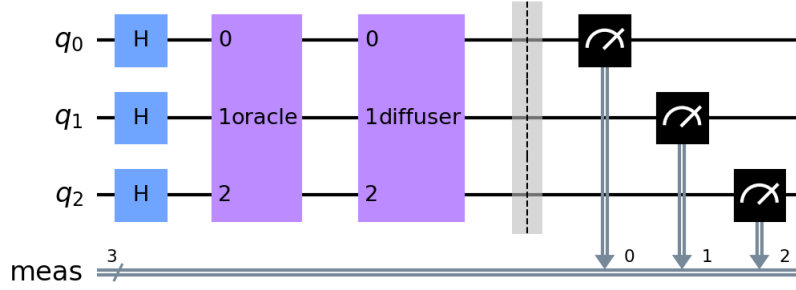


Figura 8: Exemplo algoritmo de Grover

Para 1 ou 2 valores, podemos usar a configuração acima. Mas maiores quantidade de valores, precisamos adicionar o conjunto Oracle + Diffuser k vezes, sendo $k \approx \frac{\pi}{4\sqrt{\frac{a}{2^a}}} - \frac{1}{2}$, sendo a o numero de valores marcados.

3.1.3 Diferença de conjuntos

Utilizando os phase oracles, podemos criar dois Oracles distintos com ranges de valores diferentes e sobrepor seus valores, realizando a operação de diferença de conjuntos.

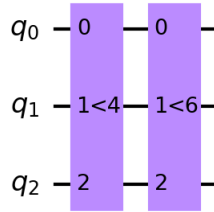


Figura 9: Exemplo de diferença de conjuntos

Nesse exemplo foi encodado no primeiro oracle o set $\{000, 001, 010, 0110\}$ e no segundo $\{000, 001, 010, 011, 100, 101\}$.

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Figura 10: Resultado da diferença de conjuntos

Como pode ser visto, apenas os valores $\{100, 101\}$ permaneceram com a fase, representando então a sobreposição delas.

3.1.4 Solução para o problema

Para a solução do problema, podemos criar em uma hash function $C(v)$ da qual recebe o path de um arquivo e retorna uma bit string respectiva. Com essa função em mãos, podemos utilizar o conjunto dos valores retornados e encoda-los em um phase oracle, agindo como uma especie de Look Up Table para os arquivos existentes na máquina.

No entanto, para ter sucesso na pesquisa, é necessário utilizar um segundo Oracle encondando $S = P - s$, sendo s o conjunto de arquivos do qual estamos procurando e S os arquivos restantes, aproveitando-se então da diferença de conjuntos para encontrar então apenas os valores desejados para a pesquisa.

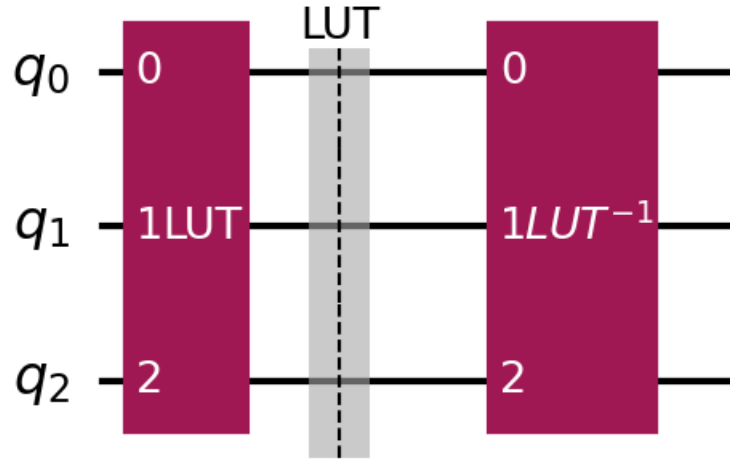


Figura 11: Diferença de conjuntos com as Look Up Tables

Sendo assim, o primeiro oracle age como o HD da marquina, marcando todos os arquivos existentes, e o segundo age como o mediador da pesquisa.

Com isso, adicionamos a Look Up Table final ao algoritmo de Grover.

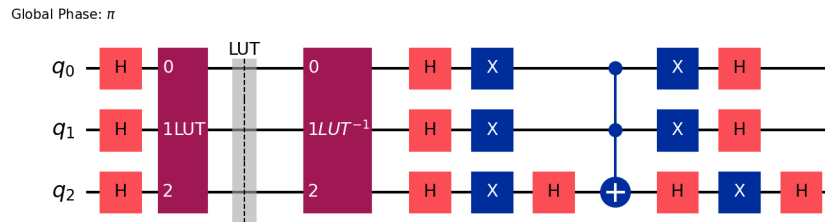


Figura 12: File explorer circuito

3.1.5 Resultados

Para um caso hipotetico de sistema completamente quântico, certamente esse é um das maneiras de encontrar arquivos em meio a todos os outros.

Contudo, como o objetivo dessa pesquisa sugere, para utilizar esse modelo em um sistema clássico tomando proveito da computação quântica, não se mostra como a melhor opção.

Para sistemas convencionais, dos quais utilizam métodos baseados em árvores, não é possível tirar qualquer proveito aqui, sendo O polinomial contra O quadrático.

Mesmo perante os testes clássicos feitos durante o desenvolvimento, esse protocolo só se mantém útil quando a pesquisa era feita de forma linear.

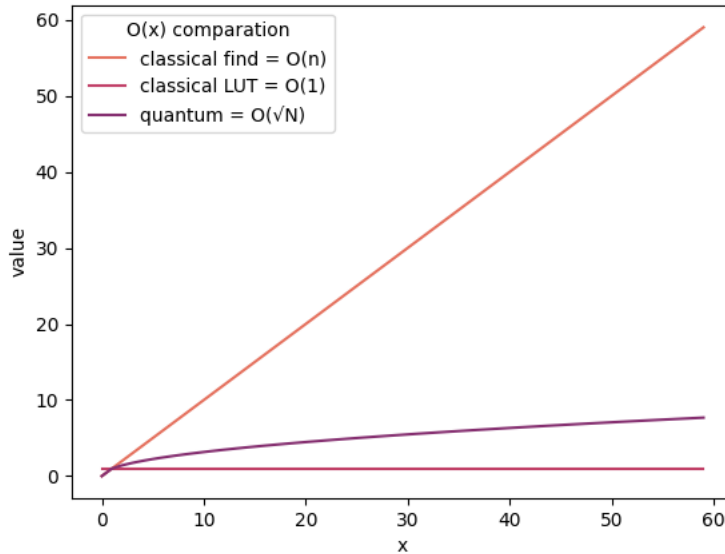


Figura 13: Comparação algoritmos usados na pesquisa

Sendo assim, o algoritmo de Grover deve ser pensando para casos do qual a versão clássica possui complexidade $\geq O(n)$.

3.2 Miles to Kilometers

3.3 Hanoi Tower

3.4 Buckshot Roulette

3.5 QRAM

Referências

- [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] Ryan O’Donnell. Lecture 5: Quantum query complexity, 09 2015.
- [3] Dave Bacon. Cse 599d -quantum computing simon’s algorithm, 2006.
- [4] Robin Kothari. An optimal quantum algorithm for the oracle identification problem. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2014.
- [5] Ryan O’Donnell. Lecture 13: Lower bounds using the adversary method, 10 2015.
- [6] Laurel Brodkorb and Rachel Epstein. The entscheidungsproblem and alan turing, 12 2019.
- [7] Sadika Amreen and Reazul Hoque. Oracle turing machines.
- [8] Subrahmanyam Kalyanasyndaram. mod04lec23 - oracle turing machines, 09 2021.
- [9] Martin Davis. Turing reducibility?, 11 2006.
- [10] Mahesh Viswanathan. Reductions 1.1 introduction reductions, 2013.
- [11] 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.
- [12] Takashi Yamakawa and Mark Zhandry. Classical vs quantum random oracles. Cryptology ePrint Archive, Paper 2020/1270, 2020. <https://eprint.iacr.org/2020/1270>.

-
- [13] Harry Buhrman, Richard Cleve, and Avi Wigderson. Quantum vs. classical communication and computation, 1998.
- [14] Javier Sanchez-Rivero, Daniel Talaván, Jose Garcia-Alonso, Antonio Ruiz-Cortés, and Juan Manuel Murillo. Some initial guidelines for building reusable quantum oracles, 2023.
- [15] Austin Gilliam, Marco Pistoia, and Constantin Gonciulea. Canonical construction of quantum oracles, 2020.
- [16] Elham Kashefi, Adrian Kent, Vlatko Vedral, and Konrad Banaszek. Comparison of quantum oracles. *Physical Review A*, 65(5), May 2002.
- [17] Niklas Johansson and Jan-Åke Larsson. Quantum simulation logic, oracles, and the quantum advantage. *Entropy*, 21(8), 2019.
- [18] William Zeng and Jamie Vicary. Abstract structure of unitary oracles for quantum algorithms. *Electronic Proceedings in Theoretical Computer Science*, 172:270–284, December 2014.
- [19] Alp Atici. Comparative computational strength of quantum oracles, 2004.
- [20] Kathiresan Sundarappan. How to build oracles for quantum algorithms, 04 2022.
- [21] 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.
- [22] 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.
- [23] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. Quantum random access memory. *Physical Review Letters*, 100(16), April 2008.
- [24] Samuel Jaques and Arthur G. Rattew. Qram: A survey and critique, 2023.
- [25] 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>.
- [26] Piotr Frackiewicz. Application of the ewl protocol to decision problems with imperfect recall, 2011.
- [27] Jens Eisert, Martin Wilkens, and Maciej Lewenstein. Quantum games and quantum strategies. *Physical Review Letters*, 83(15):3077–3080, October 1999.
- [28] Muhammad Usman. Kilometres to miles conversion — approximation of fibonacci series, 09 2019.
- [29] Lidia André. Tower of hanoi – lidia andré, 03 2021.
- [30] diptokarmakar47. How to solve the tower of hanoi problem - an illustrated algorithm guide, 01 2019.
- [31] Towers of hanoi: A complete recursive visualization, 05 2020.
- [32] GeeksforGeeks. Program for tower of hanoi, 05 2014.
- [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–wilkins–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.