Technical University of Denmark

Quantum Information

Grover's algorithm for the Travelling Salesman Problem (TSP)

Author:
Victor Niaussat

April 30, 2021

Contents

1 How Grover's algorithm works.

Grover's algorithm is a quantum algorithm for unstructured search that finds with high probability the input ω to a black box function that produces a particular output value. This black box is a function f_ω

$$f_\omega(x) = \begin{cases} 1 & \text{if } x \in \omega \\ 0 & \text{otherwise} \end{cases}$$

We can access f_ω with a subroutine (sometimes called an oracle) in the form of a unitary operator U_ω that acts as follows:

$$U_\omega|x\rangle = \begin{cases} -|x\rangle & \text{if } x \in \omega \\ |x\rangle & \text{otherwise} \end{cases}$$

Grover's algorithm incorporates a second main element : an amplitude amplification algorithm which named diffusion operator, which makes the information given by the black box usable and measurable. This algorithm is independent of the black box, and it is this procedure that requires $O(\sqrt{N})$ iterations. This diffusion operator mirrors the amplitudes around the average of the amplitudes. This has the effect of amplifying the target states, and decreasing the other states. Here is a representation of the probability amplitudes of the steps of the Grover's algorithm :

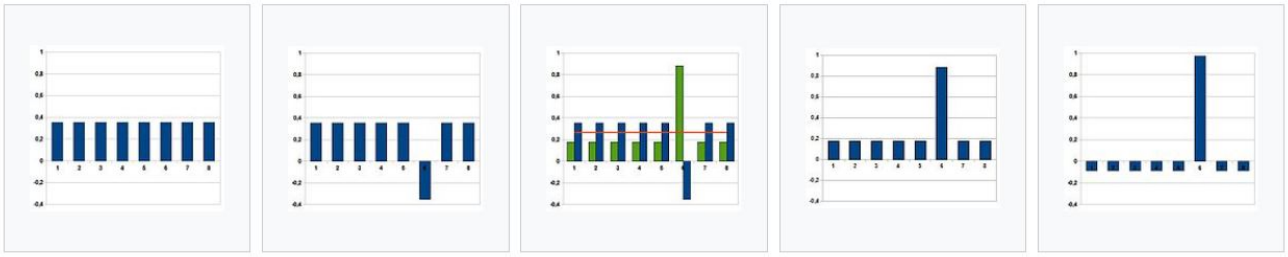
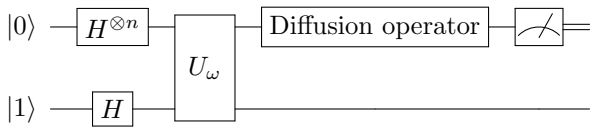
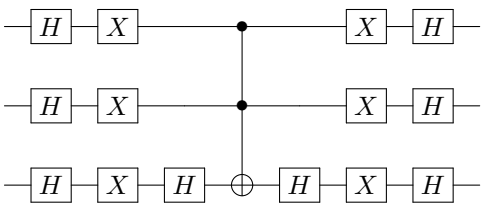


Figure 1: Representation of the probability amplitudes : ***1.*** *Initial state* ***2.*** *Application of the oracle* ***3.*** *Application of the "mirror around the mean" operator* ***4.*** *Amplified state* ***5.*** *Final state, after several more iteration*

To implement the algorithm, we need to start with n qubits $|0\rangle$ to represent $N = 2^n$ numbers and a qubit $|-\rangle$ (obtained by applying a Hadamard gate to a qubit $|1\rangle$) to have an input of the oracle represented by U_ω. Then, there is a diffusion operator which applies to the n first qubits. We need to iterate the oracle and the diffusion operator several times to amplify the probabilities of the targets. The optimal number of iterations is $\frac{\pi}{4}\sqrt{2^n}$. At the end of the algorithm, we measure the n first qubits. This is below the quantum circuit representing the Grover algorithm:



The diffusion operator is $\hat{D} = \hat{H}\hat{Z}\hat{H}$ where $\hat{H}$ is the Hadamard gate and $\hat{Z}$ the "Zero phase shift " with $\hat{Z} = 2|0\rangle\langle 0| - \hat{I}$. We can implement this operator in a quantum circuit like that (with 3 qubits) :



2 How it applies to the particular problem below.

In this assignment, we compute the Grover's algorithm for the Travelling Salesman Problem (TSP).One is given a list of cities and the distances between them,the goal is to find the shortest route which visits every city exactly once. It represents each cities by a vertex and an edge represent a road between 2 cities, weighted by the distance The following search problem using Grover's algorithm is: **Find all routes of length less than 15 km**. We consider the following TSP corresponding to the graph below:

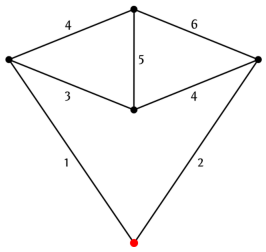


Figure 2: Graph representing the travelling salesman problem

First, we can denotes how many spanning trees in the graph above where the degree of the red vertex is one. It is the number of how many possible routes are there, starting from the mill and visiting each village exactly once. So there are 8 routes, which it labelled from 0 to 7. Hence, $8 = 2^3$ and we can use 3 qubits to labelled all of them:

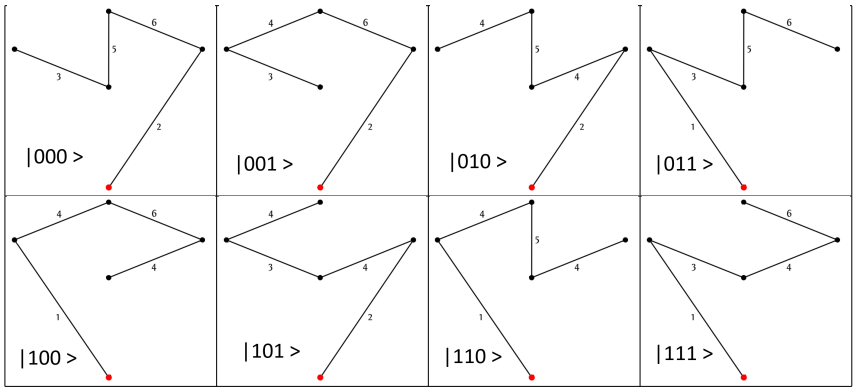


Table 1: Possible routes from the red vertex

Then,we denotes how many kilometers measures each routes in this table :

Road n°	Qubit	Total distance
0	$\|000\rangle$	16
1	$\|001\rangle$	15
2	$\|010\rangle$	15
3	$\|011\rangle$	15
4	$\|100\rangle$	15
5	$\|101\rangle$	13
6	$\|110\rangle$	14
7	$\|111\rangle$	14

Table 2: Possible routes from the red vertex

Hence, we can see that only the routes 5 , 6 and 7 have a length less than 15 km. The purpose is then to implement an oracle such as :

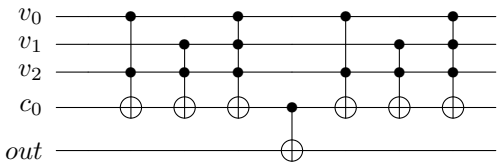
$$U|x\rangle = \begin{cases} |x\rangle & \text{if } x \in \{0,1,2,3,4\} \\ -|x\rangle & \text{if } x \in \{5,6,7\} \end{cases}$$

3 How your circuit works and how you implement it

To implement our circuit, we need to implement an oracle such as :

$$U|x\rangle = \begin{cases} |x\rangle & \text{if } x \in \{0,1,2,3,4\} \\ -|x\rangle & \text{if } x \in \{5,6,7\} \end{cases}$$

Thanks to these gates below, we can implement this oracle and have an output such that the qubit output is $|0\rangle$ when the three first qubits are not a target, and the qubit output is $|1\rangle$ if the three first qubits are a target. We need to use a fourth qubit. The first Toffle gate give transform the qubit $c_0 = |0\rangle$ to $|1\rangle$ where $v_0 = v_2 = 1$, so for the qubits $|101\rangle$ and $|111\rangle$. The second Toffle gate applies a NOT gate to the qubit c_0 if $v_1 = v_2 = 1$, so for the qubits $|110\rangle$ and $|111\rangle$. The last gate applies a NOT gate to the qubit c_0 if $v_0 = v_1 = v_2 = 1$ so for the qubits $|111\rangle$. A $CNOT$ gate changes the output qubit if $c_0 = 1$. In the end, the output is $|1\rangle$ if $|v_2 v_1 v_0\rangle \in \{5,6,7\}$. Otherwise, the output is $|1\rangle$. Then, we apply the same gate to reset c_0. This circuit can represent an oracle of this problem:



So, we can implement that in a *Qiskit* program and obtain the full circuit of the Grover's algorithm. It shows here just one iteration to see all the circuit clearly.

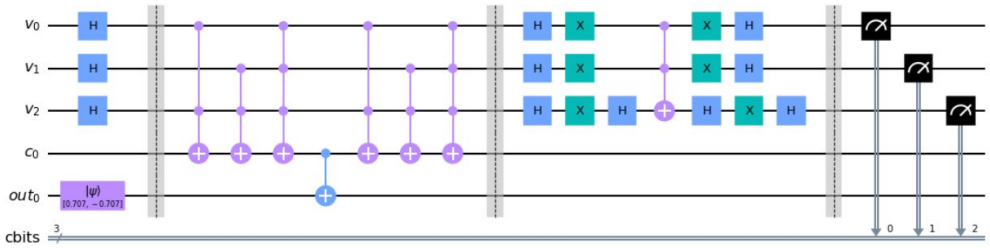


Figure 3: Grover's Algorithm's in quantum circuits with one iteration

To have the best solution, we need to implement the optimal number of iteration $n = \frac{\pi}{4}\sqrt{2^3} \approx 3$ and here are the results :

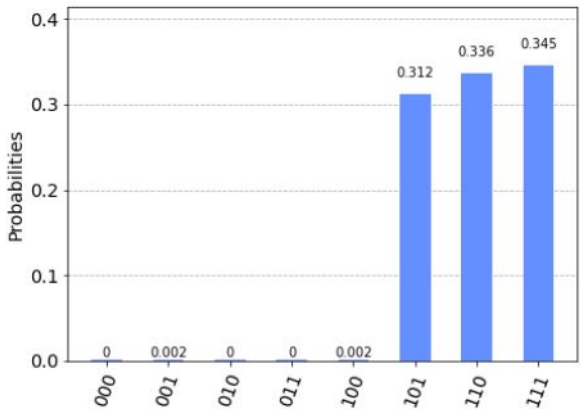


Figure 4: Probabilities of circuit's outputs

So, our Grover's algorithm can output labels of routes of length less than 15 km which are clearly here $|101\rangle = |5\rangle$,$|110\rangle = |6\rangle$ and $|111\rangle = |7\rangle$.

4 Code Qiskit

```
#initialization
import matplotlib.pyplot as plt
import numpy as np

# importing Qiskit
from qiskit import IBMQ, Aer, assemble, transpile
from qiskit import QuantumCircuit, ClassicalRegister, QuantumRegister
from qiskit.providers.ibmq import least_busy

# import basic plot tools
from qiskit.visualization import plot_histogram

def initialize_s(qc, qubits): #Apply a H-gate to 'qubits' in qc
    for q in qubits:
        qc.h(q)
    return qc

def oracle(qc):
    qc.barrier()
    qc.ccx(0,2,3) #Toffle gate controlled v0 and v2
    qc.ccx(1,2,3)  #Toffle gate controlled v1 and v2
    qc.mct(var_qubits, clause_qubits)  #Gate controlled v0,v1 and v2
    qc.cx(3,4) #CNOT gate controlled the output qubit.
    qc.ccx(0,2,3)
    qc.ccx(1,2,3)
    qc.mct(var_qubits, clause_qubits)

def diffusion(qc):
    qc.barrier()
    qc.h(N)
    qc.x(N)
    qc.h(n-1)
    qc.ccx(0,1,2)
    qc.h(n-1)
    qc.x(N)
    qc.h(N)
    qc.barrier()

n = 3

# Create separate registers to name bits
var_qubits = QuantumRegister(n, name='v')  # variable bits
clause_qubits = QuantumRegister(1, name='c')
output_qubit = QuantumRegister(1, name='out')
# bits to store clause-checks
cbits = ClassicalRegister(3, name='cbits')

# Create quantum circuit
grover_circuit = QuantumCircuit(var_qubits, clause_qubits,output_qubit,cbits)
grover_circuit.initialize([1, -1]/np.sqrt(2), output_qubit) #Initialize out to qubit |-> .
N=[i for i in range(n)]
grover_circuit = initialize_s(grover_circuit,N)

for i in range(3): #We iterrate the oracle and the diffusion operator.
    # Oracle
    oracle(grover_circuit)
    # Diffusion operator
    diffusion(grover_circuit)

grover_circuit.measure(var_qubits,cbits) #Measure of the qubits v0 v1 and v2.

sv_sim = Aer.get_backend('statevector_simulator')
qobj = assemble(grover_circuit)
result = sv_sim.run(qobj).result()
statevec = result.get_statevector()
grover_circuit.draw('mpl',fold=-1)

qasm_sim = Aer.get_backend('qasm_simulator')
qobj = assemble(grover_circuit)
result = qasm_sim.run(qobj).result()
counts = result.get_counts()
plot_histogram(counts)
```