Implementing Grover's Algorithm in *Qiskit*

Nicolás Aguado

Fundamentals of Quantum Computing — November 29, 2024

1 Introduction

The objective of this work is to implement the Quantum Grover's [1] algorithm for searching in *IBM*'s *Qiskit* framework [2].

As an starting point, the search space will be defined as a 2 qubit function. The target state that will be amplified (searched) with the algorithm will be $|01\rangle$.

2 Circuit Definition

2.1 Oracle

The following oracle leaves all qubit values unchanged and performs a phase flip only when the state is $|01\rangle$.

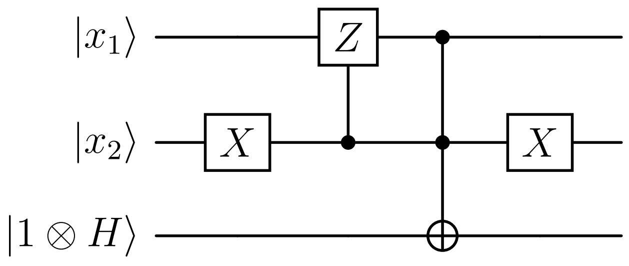

Figure 1: Oracle for the $|01\rangle$ state

In the Figure 1. it can be observed that the CZ gate is used in order to only apply the phase plip when the input state is indeed $|01\rangle$. The second qubit is inverted first with the X gate so that the phase inversion is performed on $|1\rangle$ only when $|x_2\rangle$ is $|0\rangle$. For the last qubit in this implementation it may be redundant but it is here displayed to adhere to the formal definition of the U_f.

2.2 Diffuser (Amplification Operator)

Now, to build the diffuser that performs an inversion around the mean in order to amplify the probability amplitude of the marked state, a representation of the mean inversion formula $V' = (-\mathbb{I} + 2A) \cdot V$ must be built using circuits. Thus, in the general case,

$$-\mathbb{I} + 2A \overset{[4]}{=} H^{\otimes n} \left(2|0\rangle^{\otimes n}\langle 0|^{\otimes n} - \mathbb{I}_n\right)$$

But in this concrete example, as the target state only contains 2 qubits (no ancilla qubits must be used) it can be shown [4] that the formula can be simplified to a CZ gate *sandwiched* by X gates and then by H gates on the target qubit (state preparation, reflection, inverse of state preparation).

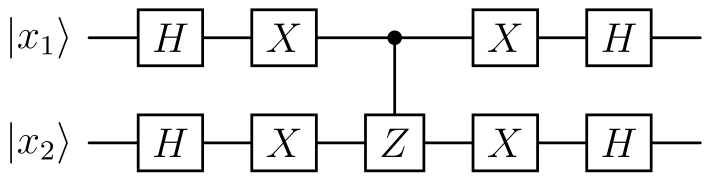

Figure 2: Diffuser circuit for the $|01\rangle$ state

The circuit shown in Figure 2 performs an inversion around the mean. In order to guarantee with enough probability that the chosen state is actually selected, the diffusion operation must be repeated at least $\frac{\pi}{4}\sqrt{2^n}$ times.

2.3 Complete Circuit

Combining all the partial circuits together, the complete diagram is shown in Figure 3.

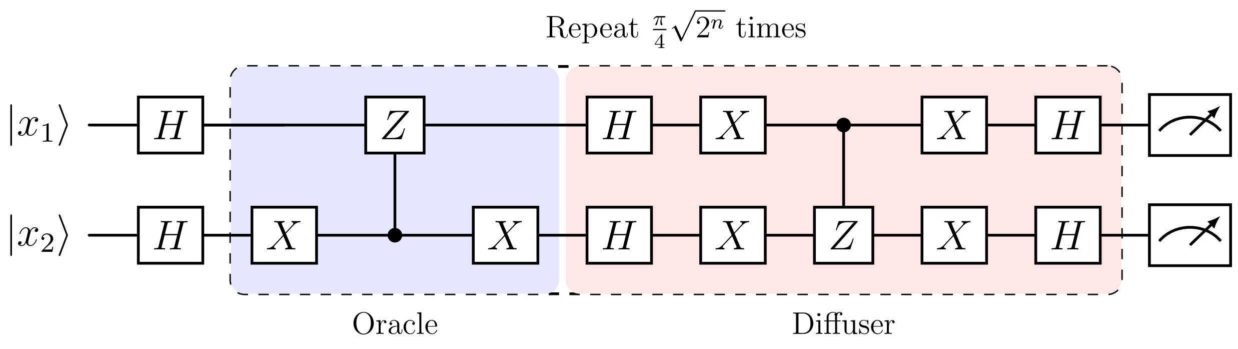

Figure 3: Grover's Algorithm for Target $|01\rangle$

Both qubits are put into superposition by the initial Hadamard gates. The oracle then marks with a phase inversion the correct state, and the diffuser will amplify the probability amplitude of said marked state. The oracle+diffuser operations will repeat enough times so that the probability of both qubits of collapsing to the target state is high enough.

Also, it has to be noted that the represented circuit is not complete, but rather the qubit used for indicating the function's result in the oracle has been simplified (represented by $1 \otimes H$ in Figure 1.).

3 Qiskit Implementation

3.1 Circuit

The represented circuit defined by Figure 3. is implemented in *Qiskit* as follows.

```
GroverQiskitNAguado.ipynb
```

```
def grovers_circuit():
 circuit = QuantumCircuit(2, 2)

 circuit.h([0,1])

 iterations = int((np.pi/4) * np.sqrt(2**2))
 for _ in range(iterations):
  circuit.x(1)
  circuit.cz(0,1)
  circuit.x(1)

  circuit.h([0,1])
  circuit.x([0,1])
  circuit.cz(1,0)
  circuit.x([0,1])
  circuit.h([0,1])

 circuit.measure_all()

return circuit
```

As a side note, in the code, the different sections of the algorithm are differentiated by spaces, each corresponding to one of the previously exposed parts of said algorithm.

3.2 Simulation Code

3.2.1 *AER* Simulator

Grover's algorithm can be run using the *AER* simulator as follows

```
GroverQiskitNAguado.ipynb
```

```
circuit = grovers_circuit()

simulator = AerSimulator()
sampler = Sampler(simulator)
job = sampler.run([circuit],  shots=1000)

result = job.result()[0]

cc = result.data.meas.get_counts()
print(f"Counts for the output register: {counts}")
plot_histogram(cc, title='1000 Grover Runs Results')
```

The circuit is first transpiled and then run 1000 times using the simulator. Then, the results are plotted using a histogram.

3.2.2 Quantum Machine Run

An alternative method of running the algorithm is using a real quantum machine from *IBM*. The code to do so is as follows

GroverQiskitNAguado.ipynb

```
IBM_TOKEN = 'TOKEN_HERE'
service = QiskitRuntimeService(channel="ibm_quantum",
     token=IBM_TOKEN)

# Optimize problem for quantum execution.
circuit = grovers_circuit()
backend = service.least_busy(operational=True,
     simulator=False)
pm = generate_preset_pass_manager(backend=backend,
     optimization_level=1)
isa_circuit = pm.run(circuit)

# 3. Execute using the Sampler primitive
sampler = Sampler(mode=backend)
sampler.options.default_shots = 1000
job = sampler.run([isa_circuit])
print(f"Job ID is {job.job_id()}")
pub_result = job.result()[0]
ck = pub_result.data.meas.get_counts()
print(f"Counts for the meas output register: {ck}")
plot_histogram(ck, title='1000 Grover Runs Results')
```

It has to be noted that an API key with at least 2s of available computing time in the `ibm_quantum` channel must be provided [3]. Again, like in the *AER* simulator, the algorithm is transpiled, optimized and being run 1000 times.

3.3 Measurement Analysis and Results

Two experiments were done. One using the AER simulation and another in an *IBM* Quantum machine.

For the first experiment (using the simulator), we can observe in the results in Figure 4. that the algorithm is collapsing to the correct state in all cases (100%).

For the second experiment (using an *IBM* machine), we can observe in Figure 5. that the algorithm is collapsing to the correct state in most cases (93.4%).

The results from the two experiments highlight the difference in performance between an idealized quantum simulation environment and a real quantum hardware implementation. The perfect outcome using the simulator underscores the difference of noise, gate errors, and decoherence effects versus a real quantum computer. The perfect accuracy serves as a baseline, showcasing the algorithm's potential under ideal conditions.

The performance gap observed on the IBM Quantum machine highlights the challenges of implementing quantum algorithms on current noisy intermediate-scale quantum (NISQ)

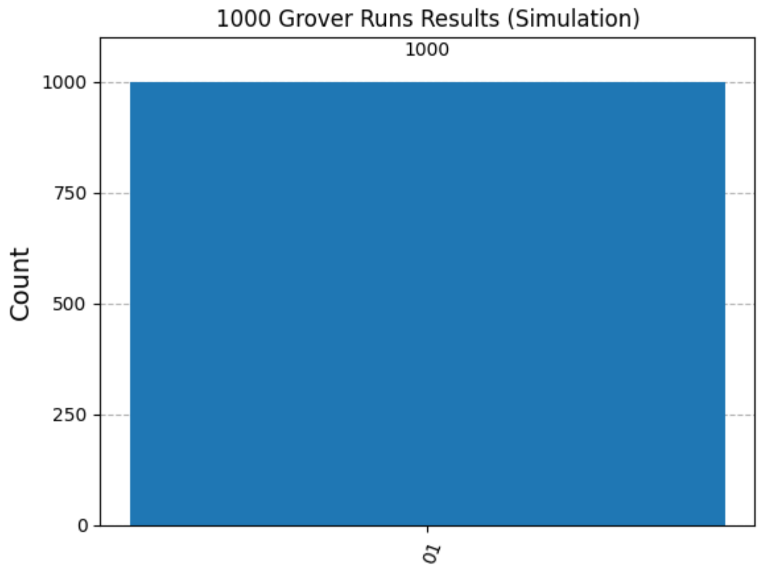

Figure 4: Simulator Results

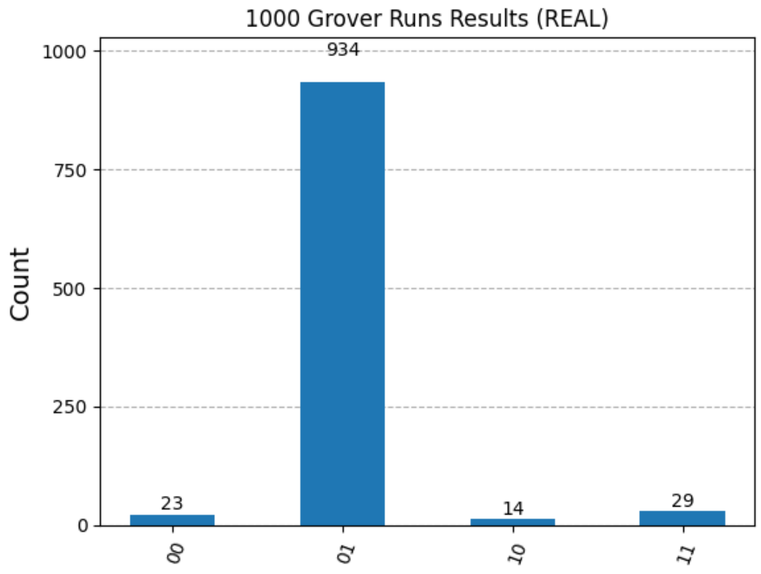

Figure 5: *IBM* System Results

devices. Despite the 6.6% reduction in success rate, the real quantum hardware results are promising, as a high success rate like 93.4% demonstrates the viability of Grover's algorithm in practical applications.

In summary, the observed results reflect the trade-offs between algorithm complexity and hardware limitations.

References

[1] Grover, L.K.: A fast quantum mechanical algorithm for database search. In: Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing. p. 212–219. STOC '96, Association for Computing Machinery, New York, NY, USA (1996). https://doi.org/10.1145/237814.237866, https://doi.org/10.1145/237814.237866

[2] IBM: IBM Qiskit Framework. https://www.ibm.com/quantum/qiskit, [Online; accessed 19-November-2024]

[3] IBM: IBM Quantum Platform. https://quantum.ibm.com/, [Online; accessed 19-November-2024]

[4] Nielsen, M.A., Chuang, I.L.: Quantum Computation and Quantum Information: 10th Anniversary Edition. Cambridge University Press, USA, 10th edn. (2011)