Grover's algorithm

Ioannis Savvaidis

1 General Analysis of Grover's Algorithm

Grover's algorithm provides a polynomial speedup for unstructured database search. The items of the database are $N = 2^n$, indexed with $i \in \{0, N-1\}$. The equal distributed probability for each query to identify the correct answer is $1/N$. This can be vary depending on the random "lucky" sequence, and an average probability of an answered query can be defined as:

$$\sum_{i \in \{0,N-1\}} \frac{1}{N} \times i = \frac{1}{N} \times \frac{(N+1)N}{2} \approx \frac{N}{2}$$

The complexity in the classical algorithm is $O(N)$, and Grover improved it to $O(\sqrt{N})$ by using quantum parallelism and interference with n-qubits with equal distributed probability. The oracle implements a function that detects if the item is found during the search. We denote the expected DB Index as x_i.

$$f : \{0,1\}^n \to \{0,1\}$$

$$f(x) = \begin{cases} 1, & \text{if } x = x_i, \text{found} \\ 0, & \text{if } x \neq x_i, \text{not found} \end{cases}$$

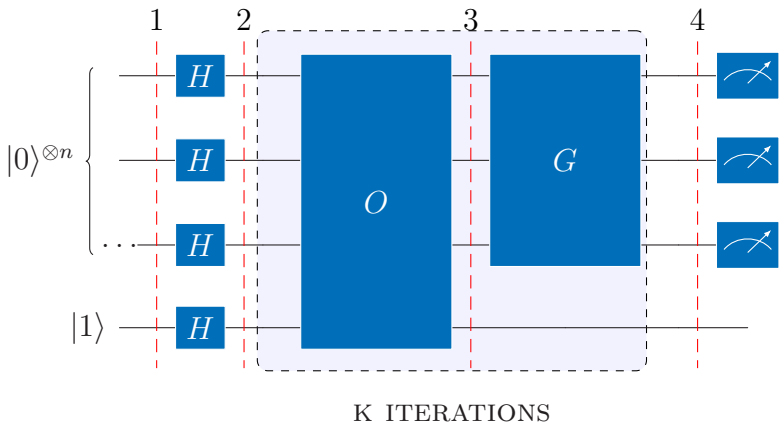


Figure 1: Grover's Algorithm Circuit

Grover's circuit (Fig. 1) consists of two operators executed sequentially in k iterations. The first operator is the oracle operator $\hat{O}$, which controls the conditional phase shift with phase kickback, based on $f(x)$ evaluation simultaneously with quantum parallelism. Next, the G operator amplifies the probability amplitude of the founded query encoded in the shifted phase of the respective state, representing the DB index. Therefore, the operator $\hat{O}$ marks the desired index, and G amplifies it to be measured.
Both are projection operators that shift the register state to the desired vector, resulting in the desired state $|x_i\rangle$.

The linear transformation P for the projection of a vector $\vec{v}$ onto a line, where the line is represented by a vector $\vec{u}$, is:

$$P_u(v) = (uu^T)v$$

In Hilbert space, the Quantum States are unit vectors of the complex vector space, the Projection operator onto a vector is Hermitian, and projecting a state to another state is an orthonomal projection, shifting the vector to an angle θ:

$$P_u = P_u^2 = u \cdot u^* = |u\rangle\langle u|$$

The reflection operator extends the projection by reflecting the vector across an axis of reflection represented by another state or a subspace. The vector is shifted by 2θ, and the operator is:

$$F_u = 2|u\rangle\langle u| - I_u$$

With these operators, we can project or reflect the register's state onto a vector subspace spanned by one or multiple states. A hyperplane can represent and abstract the vector subspace.

1.0.1 Phase Shift with Oracle Operator

Prior to the Oracle, all qubits are set in superposition. The oracle qubit $|y\rangle$ is initialized in $|1\rangle$ to achieve the phase kickback:

$$|q_1\rangle = |0\rangle^{\otimes n} \otimes |1\rangle$$

$$|q_2\rangle = H^{\otimes n}|x\rangle \otimes |-\rangle$$

Then the oracle acts and the new state is:

$$|q_3\rangle = \hat{O}|x\rangle$$

Omitting the Oracle's qubit in $|-\rangle$, the register states will be conditionally phase shifting due to $f(x)$. Therefore, the summation on $|x\rangle^{\otimes n}$ will contain phases on states depending on:

$$phase = \begin{cases} 1, & \text{if } f(x) = 0, \text{ (not found)} \\ -1, & \text{if } f(x) = 1, \text{ (found), } x = x_i. \end{cases}$$

The Oracle operator is defined as:

$$\hat{O} = \hat{I} - 2|x_i\rangle\langle x_i|$$

This operator acts as a negative reflection operator.

$$\hat{O}|x\rangle \to \begin{cases} I|x\rangle - 2|x_i\rangle\langle x_i|x\rangle = |x\rangle & \text{if } x \neq x_i, \\ I|x_i\rangle - 2|x_i\rangle\langle x_i|x_i\rangle = -|x_i\rangle & \text{if } x = x_i. \end{cases}$$

In this step, the Oracle *marks* the searched element based on the query by adding a phase π. However, the probability remains the same.

The new state can be expressed with $f(x)$, as :

$$|q_3\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (-1)^{f(x)}|x\rangle$$

We can decompose the register's state as:

$$|q_3\rangle = \alpha|\text{FALSE}\rangle + \beta|\text{TRUE}\rangle$$

$$\sum |\psi\rangle \to \begin{cases} |\text{FALSE}\rangle & \text{if } x \neq x_i, \\ |\text{TRUE}\rangle & \text{if } x = x_i, \end{cases}$$

The quantum register vector space in $\mathbb{C}^N$ decomposition into two vector subspaces has the following properties. Since all vectors are unit vectors and states, these subspaces will be orthogonal complements of each other.
Let the subspace V spanned by $|x_i\rangle$ with dim(V)=1, and the subspace S spanned by all other states with dimension dim(S)=$N-1$, which is the orthogonal complement of V:

$$S \cap V = \{0\}, \quad S \perp V, \quad \mathbb{C}^N = S \oplus V$$

Any state of the register can be expressed as a linear combination of both subspaces with:

$$< x_\perp|x_i >= 0, \quad < x_\perp|x_i >= 1, \quad < x_\perp|x_\perp >= 1 \quad \Big| \quad x_\perp \in V, x_i \in S$$

These subspaces can form a new orthonomal basis set:

$$|S\rangle = \frac{1}{\sqrt{N-1}} \left(\sum_{x \in \{0,1\}^n \setminus x_i} |x_\perp\rangle \right) \qquad |V\rangle = \frac{1}{\sqrt{N}} |x_i\rangle$$

Rewriting the state needs amplitude normalization. The basis $|V\rangle$ spanned by $|x_i\rangle$, so the amplitude is the same, but the basis $|S\rangle$ spanned by all other states.

$$|\alpha|^2 = 1 - |\beta|^2 = 1 - \left| -\frac{1}{\sqrt{N}} \right|^2 = \sqrt{1 - \frac{1}{N}} = \sqrt{\frac{N-1}{N}}$$

Therefore:

$$|q_3\rangle = \sqrt{\frac{N-1}{N}} |S\rangle - \frac{1}{\sqrt{N}} |x_i\rangle$$

Fig. 2 depicts the state shift after the oracle operator reflection relative to x_i. Since there was a negative reflection across the axis of reflection in V, the state reflected across the S vector subspace perpendicular to V. The hyperplane of S, abstracts the one dimension of the ambient space and shifts the perspective to hide all the dimensions of S into a perpendicular line relative to V.

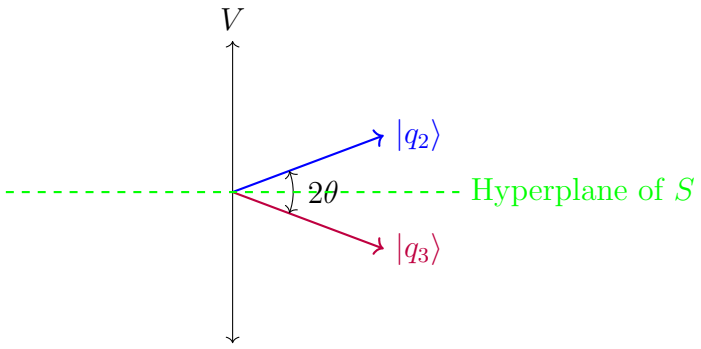


Figure 2: Reflection of state $|q_3\rangle$ after $\hat{O}|q_2\rangle$

Fig. 3 shows the expanded circuit of Oracle. The circuit consists of X and CNOT gates. The first step is the encoding of the search number. To encode it, we apply an I gate for $|1\rangle$ and an X gate for $|0\rangle$ across all qubits, of the control register. The next step is to add a CNOT with all qubits as controls, and the oracle qubit as the target. The third step is to "close" the encoding with the gate I for $|1\rangle$ and X for $|0\rangle$. The circuit in figure, encodes $|110\rangle$.

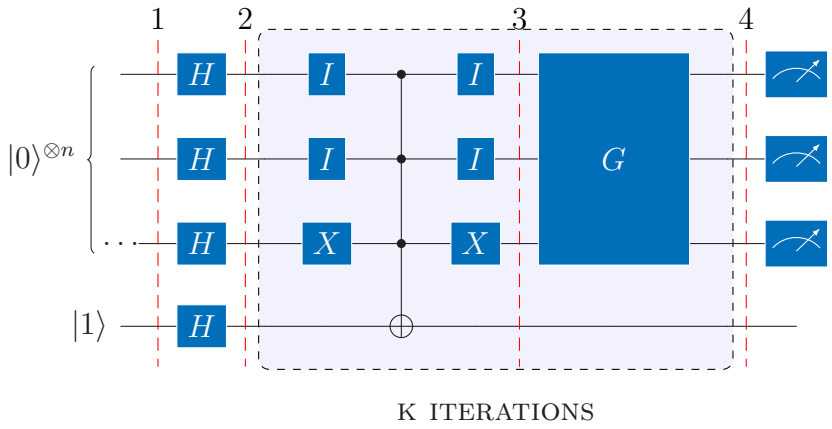


Figure 3: Oracle Expanded Circuit

1.0.2 Amplitude Amplification with Grover Diffusion Operator

The operator $\hat{G}$ amplifies the desired state, which was flipped in phase by the oracle. The amplitude increases by $\frac{2}{\sqrt{N}}$ in each iteration. After $O(\sqrt{N})$ steps, the probability amplitude at the desired state will be near 1.
Hence, the number of required iterations k of $\hat{O}$ and $\hat{G}$ is approximately:

$$\frac{\pi}{4}\sqrt{N} - 0.5$$

The Grover operator $\hat{G}$ is given by:

$$\hat{G} = 2|q_2\rangle\langle q_2| - \hat{I}$$

The operator $\hat{G}$ reflects the state after $\hat{O}$ action. Thus, it reflects the state $|q_3\rangle$ onto $|q_2\rangle$. This reflection increases the amplitude of the state spanning V and decreases the states spanning S.

$$|q_4\rangle = \hat{G}|q_3\rangle = (2|q_2\rangle\langle q_2| - \hat{I})|q_3\rangle = 2|q_2\rangle\langle q_2|q_3\rangle - |q_3\rangle \tag{1}$$

The inner product $\langle q_2|q_3\rangle$ is:

$$\langle q_2|q_3\rangle = \left(\frac{1}{\sqrt{N}}\sum_{x\in\{0,1\}^n}\langle x|\right)\left(\sqrt{\frac{N-1}{N}}|S\rangle - \frac{1}{\sqrt{N}}|x_i\rangle\right)$$

$$= \left(\frac{1}{\sqrt{N}}\sum_{x\in\{0,1\}^n}\langle x|\sqrt{\frac{N-1}{N}}\frac{1}{\sqrt{N-1}}\sum_{y\in\{0,1\}^n\setminus x_i}|y\rangle\right) - \frac{1}{\sqrt{N}}\left(\frac{1}{\sqrt{N}}\sum_{x\in\{0,1\}^n}\langle x|x_i\rangle\right) = \frac{N-2}{N}$$

The first part forms an inner product of all states, with the states span S, $\setminus x_i$. Thus, all N-1 inner products will be 1, and the summation will be N-1 times 1. The second part forms an inner product of all states with the x_i state, which will be 0 for N-1 times and 1 for x_i with itself.
Equation (1) becomes:

$$|q_4\rangle = 2|q_2\rangle\left(\frac{N-2}{N}\right) - |q_3\rangle$$

$$= 2\left(\frac{N-2}{N}\right)\left(\frac{1}{\sqrt{N}}\sum_{x\in\{0,1\}^n}|x\rangle\right) + \frac{N-2}{\sqrt{N}}|x_i\rangle - \sqrt{\frac{N-1}{N}}|S\rangle$$

Removing x_i from the first summation:

$$|q_4\rangle = 2\left(\frac{N-2}{N}\right)\left(\frac{1}{\sqrt{N}}\sum_{x\in\{0,1\}^n\setminus x_i}|x\rangle + \frac{1}{\sqrt{N}}|x_i\rangle\right) + \frac{N-2}{\sqrt{N}}|x_i\rangle - \sqrt{\frac{N-1}{N}}|S\rangle$$

Replacing $|S\rangle$:

$$|q_4\rangle = 2\left(\frac{N-2}{N}\right)\left(\frac{1}{\sqrt{N}}\sum_{x\in\{0,1\}^n\setminus x_i}|x\rangle + \frac{1}{\sqrt{N}}|x_i\rangle\right) + \frac{N-2}{\sqrt{N}}|x_i\rangle - \sqrt{\frac{N-1}{N}}\frac{1}{\sqrt{N-1}}\sum_{x\in\{0,1\}^n\setminus x_i}|x\rangle$$

Factoring out the common parts:

$$|q_4\rangle = \frac{3N-4}{N\sqrt{N}}|x_i\rangle + \frac{N-4}{N\sqrt{N}}\sum_{x\in\{0,1\}^n\setminus x_i}|x\rangle$$

For a quick verification, if N=4:

$$|q_4\rangle = 1|x_i\rangle + 0\sum_{x\in\{0,1\}^n\setminus x_i}|x\rangle$$

We can observe that the probability amplitude of x_i is more significant than every state in S. Thus, the $\hat{G}$, amplified it. Fig. 4 shows the new reflection of state $|q_4\rangle$.

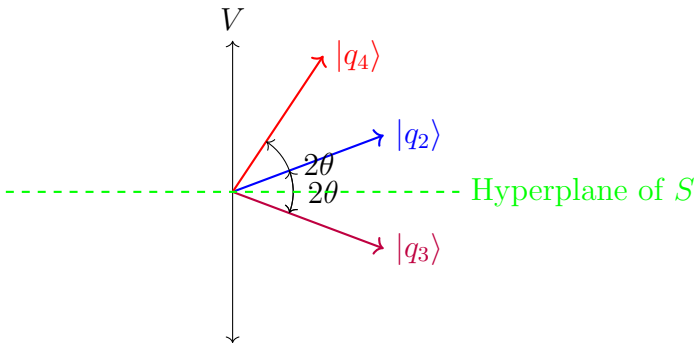


Figure 4: Reflection of state $|q_4\rangle$ after $\hat{G}|q_3\rangle$

The amplitude amplification circuit consists of H, X, and CZ gates. The first step applies a set of H and X gates to all control qubits except the oracle qubit. The second step applies a CZ gate with control on all $n-1$ qubits, and targets the n-th qubit, not the oracle qubit. The last step applies a set of H and X gates again to all controll n qubits. As in the Deutsch algorithm, we measure all n qubits in the control or upper register.

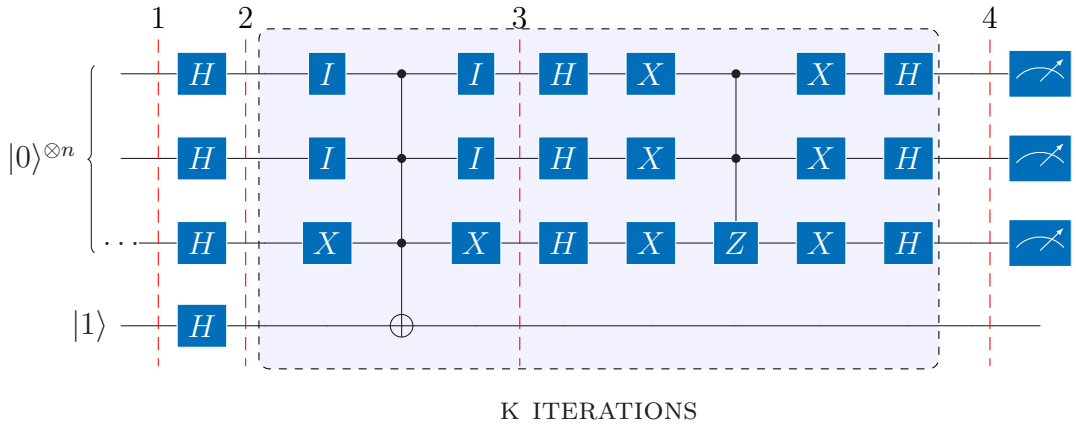
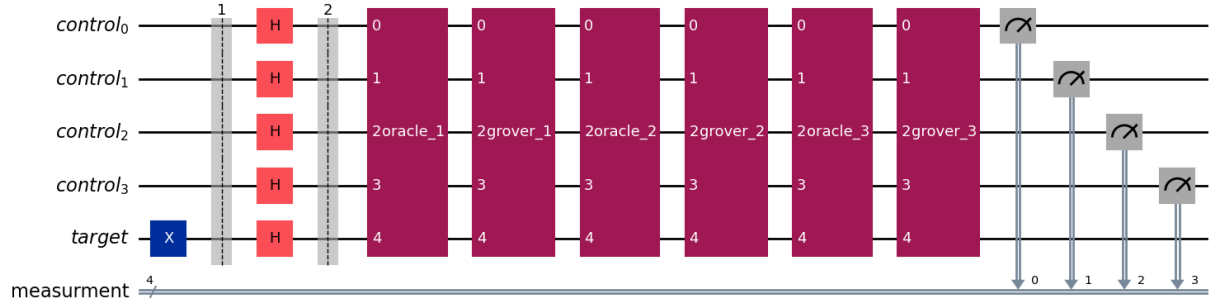


Figure 5: Grover Expanded Circuit

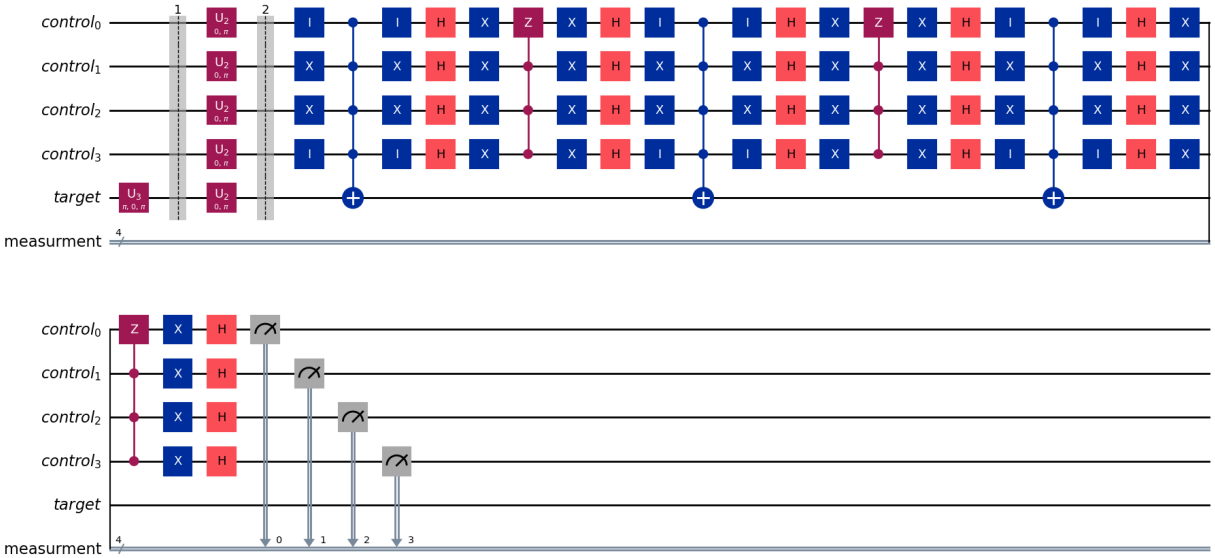
1.1 Qiskit Simulation of Grover Algorithm

Fig. 6, depicts the circuit and the results.

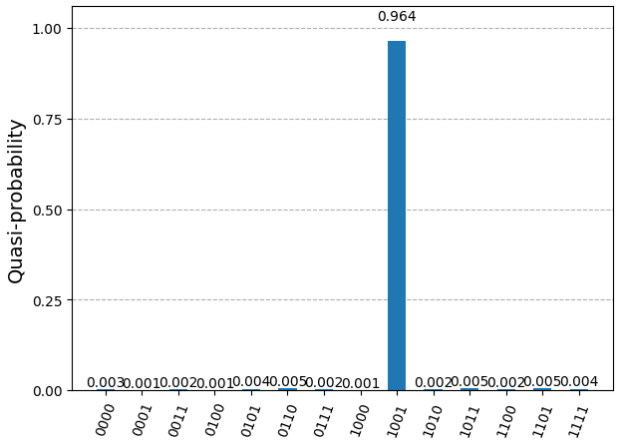
Figure 6: Grover Algorithm on Simulation



Circuit



Circuit Decomposed

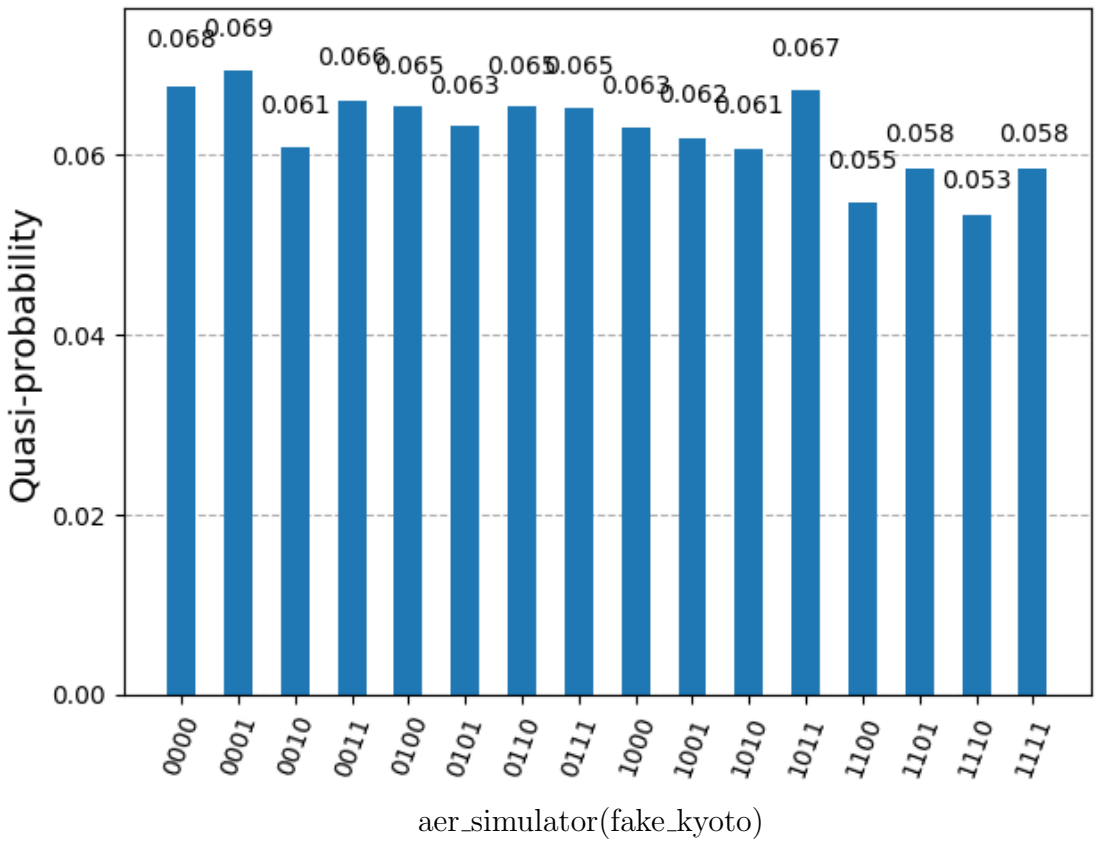


Results

Executing on real Quantum Computer

During experiments, I faced a few issues executing the circuit on Fake and Real Backends. The code executed on **ibm_kyoto**, due to immediate availability, and simulated on **fake_kyoto** backend. The real_kyoto and fake_kyoto results were always noisy, and the final measurement Fig. 7 had no clear probability of the expected outcome, as found in the ideal simulation. Questioning about my circuit optimization for the real backends, I experimented with the official Grover operator package from qiskit, and executed it in various fake backends. Indeed, Kyoto was unreliable again, even for this circuit, leading me to change the backend. FakeSherbrooke and FakeBrisbane were reliable, and I could verify the presented result from qiskit tutorial about Grover. Furthermore, I executed my circuit in these backends and got less noisy and more reliable results, so I could consider my circuit to be okay.

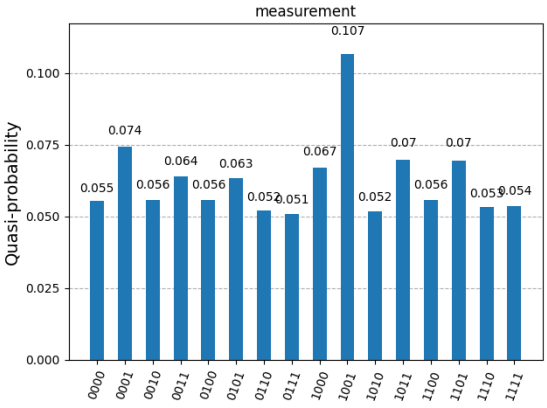
Figure 7: Results/Counts



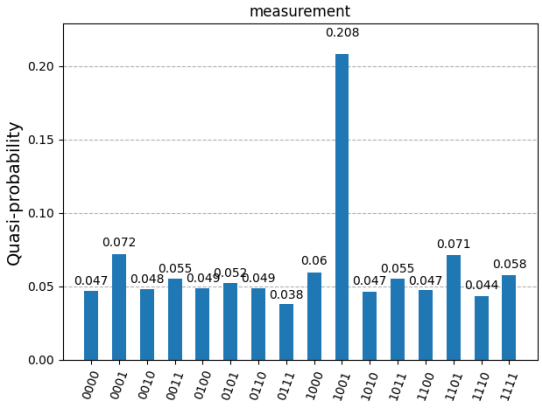
Analyzing further and using only FakeSherbrooke, before executing on real ibm_sherbrooke, I've noticed that using the **scheduling_method='alap'** or **scheduling_method='asap'**, during transpilation, I could slight observe the expected result, but with more noisy and

less reliable results Fig 8, leading me to a question on when and how to use these options. However, these transpilation options affect the reliability of the final result, and I left them empty. I noticed a more reliable result, with accepted noise Fig. 8.

Figure 8: aer_simulator(fake_sherbrooke)



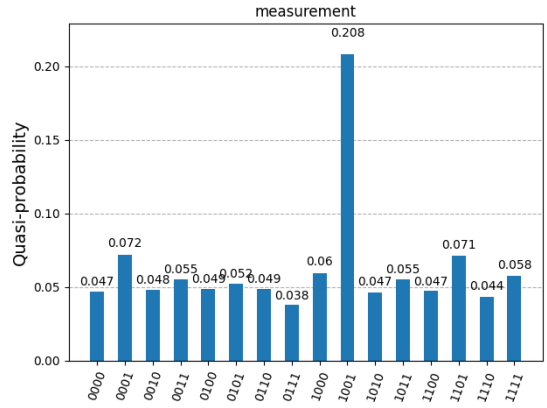
with scheduling_method=asap—alap



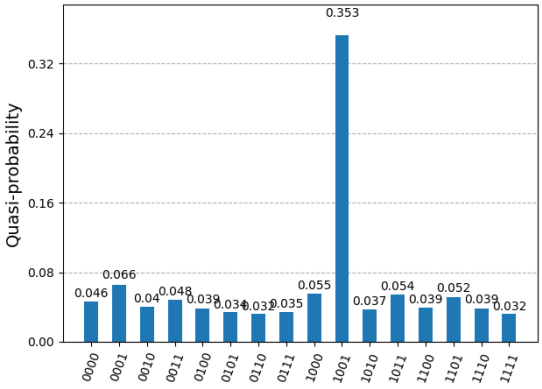
without scheduling_method=asap—alap

Extending the debugging further, I compared my circuit on the same backend with the Grover Operator-ready circuit from the qiskit tutorial, searching for the number 1001, and I noticed better clarity on the tutorial circuit. I assume that the gates would be efficiently reduced and optimized.

Figure 9: Grover Circuits results



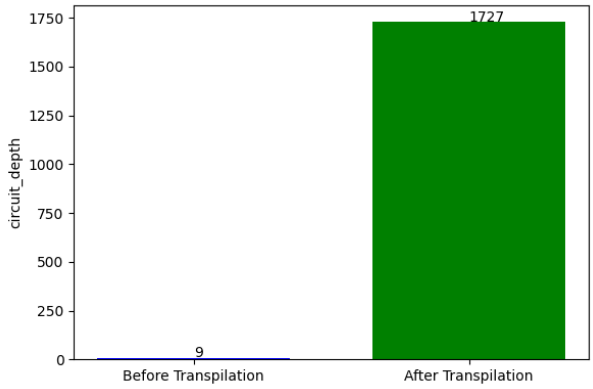
My Circuit



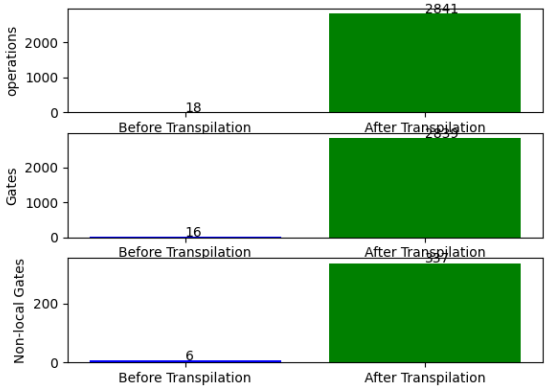
Grover Qiskit Tutorial

Since the result from my circuit was acceptable from FakeSherbrooke backend, I executed the circuit on real ibm_sherbrooke backed. The following section has the circuit and job metrics.

Figure 10: Circuit Metrics on ibm_sherbrooke

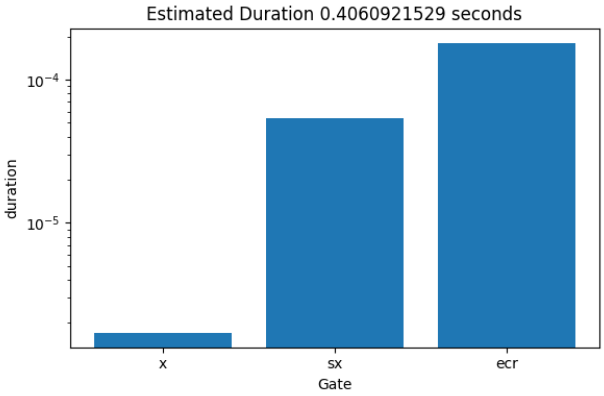


Circuit Depth)



Operations

Figure 11: Estimated Duration



Duration