2 GROVER'S ALGORITHM

2.1 Problem definition and background

Grover's algorithm as initially described [52] enables one to find (with probability $> 1/2$) a specific item within a randomly ordered database of N items using $O(\sqrt{N})$ operations. By contrast, a classical computer would require $O(N)$ operations to achieve this. Therefore, Grover's algorithm provides a quadratic speedup over an optimal classical algorithm. It has also been shown [14] that Grover's algorithm is optimal in the sense that no quantum Turing machine can do this in less than $O(\sqrt{N})$ operations.

While Grover's algorithm is commonly thought of as being useful for searching a database, the basic ideas that comprise this algorithm are applicable in a much broader context. This approach can be used to accelerate search algorithms where one could construct a "quantum oracle" that distinguishes the needle from the haystack. The needle and hay need not be part of a database. For example, it could be used to search for two integers $1 < a < b$ such that $ab = n$ for some number n, resulting in a factoring algorithm. Grover's search in this case would have worse performance than Shor's algorithm [93, 94] described below, which is a specialised algorithm to solve the factoring problem. Implementing the quantum oracle can be reduced to constructing a quantum circuit that flips an ancillary qubit, q, if a function, $f(\mathbf{x})$, evaluates to 1 for an input $\mathbf{x}$. We use the term *ancilla* or *ancillary qubit* to refer to some extra qubits that are used by the algorithm.

The function $f(\mathbf{x})$ is defined by

$$f(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} = \mathbf{x}^* \\ 0 & \text{if } \mathbf{x} \neq \mathbf{x}^* \end{cases} \tag{32}$$

where $\mathbf{x} = x_1 x_2 \ldots x_n$ are binary strings and $\mathbf{x}^*$ is the specific string that is being sought. It may seem paradoxical at first that an algorithm for finding $\mathbf{x}^*$ is needed if such a function can be constructed. The key here is that $f(\mathbf{x})$ need only recognize $\mathbf{x}^*$ – it is similar to the difference between writing down an equation and solving an equation. For example, it is easy to check if the product of a and b is equal to n, but harder to factor n. In essence, Grover's algorithm can invert an arbitrary function with binary outputs, provided we have a quantum oracle that implements the function. Grover's algorithm has been used, with appropriate oracles, to solve problems like finding triangles in a graph [72], finding cycles [28], and finding maximal cliques [109]. For the analysis of Grover's algorithm, the internals of the oracle is typically considered a black-box. Often, the oracle operator for the problem at hand has to be constructed as a quantum circuit. But, keep in mind that an inefficient oracle construction can nullify any practical advantages gained by using Grover's search.

Here we implement a simple instance of Grover's algorithm. That is, the quantum oracle we utilize is a very simple one. Let $\mathbf{x} = x_1 x_2$ and we wish to find $\mathbf{x}^*$ such that $x_1^* = 1$ and $x_2^* = 1$. While finding such an x^* is trivial, we don a veil of ignorance and proceed as if it were not. This essentially means that our function $f(\mathbf{x})$ is an AND gate. But AND gate is not reversible and cannot be a quantum gate. However the Toffoli gate, that was introduced in the previous section, is a reversible version of the classical AND gate. The Toffoli gate takes three bits as input and outputs three bits. The first two bits are unmodified. The third bit is flipped if the first two bits are 1. The unitary matrix corresponding to the Toffoli gate can be found in Table 1. In other words, the Toffoli gate implements our desired quantum oracle where the first two inputs are x_1 and x_2 and the third bit is the ancillary bit, q. The behavior of the oracle in general is $|\mathbf{x}\rangle\,|q\rangle \rightarrow |\mathbf{x}\rangle \left|f(\mathbf{x}) \bigoplus q\right\rangle$, where $\bigoplus$ is the XOR operation . Here we will only discuss the case where $\mathbf{x}^*$ is unique. Grover's algorithm can also be used to search for multiple items in a database.

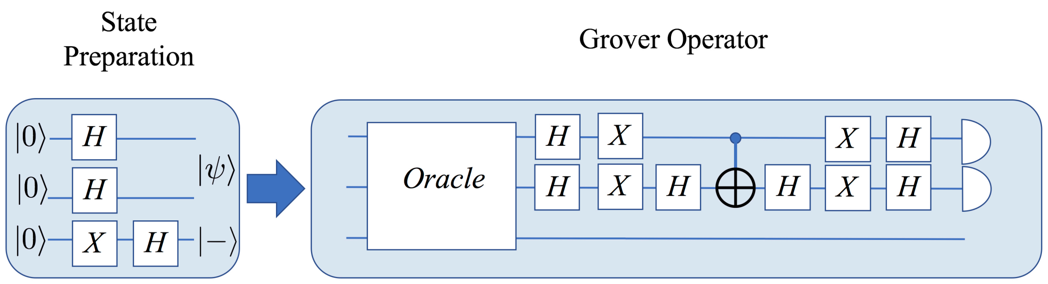

Fig. 6. A schematic diagram of Grover's algorithm is shown. Note that in this case, one application of the Grover operator is performed. This is all that is necessary when there are only two bits in $\mathbf{x}$, but the Grover operator should be applied repeatedly for larger problems.

2.2 Algorithm description

Here we present a brief introduction to Grover's algorithm. A more detailed account can be found in Nielsen and Chuang [77]. Let N be the number of items (represented as bit strings) amongst which we are performing the search. This number will also be equal to the dimension of the vector space we are working with. An operator, called the Grover operator or the diffusion operator, is the key piece of machinery in Grover's algorithm. This operator is defined by

$$G = (2\,|\psi\rangle\,\langle\psi| - I)O \tag{33}$$

where $|\psi\rangle = \frac{1}{\sqrt{N}}\sum_i |i\rangle$ is the uniform superposition over all the basis states and O is the oracle operator (see Fig. 6 for a representation of this operator in the case where $\mathbf{x}$ consists of 2 bits). The action of $(2\,|\psi\rangle\,\langle\psi| - I)$ on an arbitrary state, given by $\sum_i a_i\,|i\rangle$, when decomposed over the basis states is,

$$(2\,|\psi\rangle\,\langle\psi| - I)\sum_i a_i\,|i\rangle = \sum_i (2\,\langle a\rangle - a_i)\,|i\rangle \tag{34}$$

where $\langle a\rangle = \frac{\sum_i a_i}{N}$ is the average amplitude in the basis states. From Eq. (34) one can see that the amplitude of each $|i\rangle$-state (a_i) is flipped about the mean amplitude.

In order to use the Grover operator to successfully perform a search, the qubit register must be appropriately initialized. The initialization is carried out by applying a Hadamard transform to each of the the main qubits $(H^{\otimes n})$ and applying a Pauli X transform followed by a Hadamard transform (HX) to the ancilla. This leaves the main register in the uniform superposition of all states, $|\psi\rangle$, and the ancilla in the state $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$. After performing these operations, the system is in the state $|\psi\rangle\,\frac{|0\rangle-|1\rangle}{\sqrt{2}}$. Using Eq. (34), we can now understand how the Grover operator works. The action of the oracle operator on $|\mathbf{x}^*\rangle\,\frac{|0\rangle-|1\rangle}{\sqrt{2}}$ reverses the amplitude of that state

$$O\,|\mathbf{x}^*\rangle\,\frac{|0\rangle - |1\rangle}{\sqrt{2}} \rightarrow |\mathbf{x}^*\rangle\,\frac{\left|f(\mathbf{x}^*)\bigoplus 0\right\rangle - \left|f(\mathbf{x}^*)\bigoplus 1\right\rangle)}{\sqrt{2}} = |\mathbf{x}^*\rangle\,\frac{|1\rangle - |0\rangle}{\sqrt{2}} = -\,|\mathbf{x}^*\rangle\,\frac{|0\rangle - |1\rangle}{\sqrt{2}} \tag{35}$$

A similar argument shows that all other states are unmodified by the oracle operator. Combining this with Eq. (34) reveals why the Grover operator is able to successfully perform a search. Consider what happens on the first iteration: The oracle operator makes it so that the amplitude of $|\mathbf{x}^*\rangle$ is below $\langle a\rangle$ (using the notation of Eq. (34)) while all the other states have an amplitude that is slightly above $\langle a\rangle$. The effect of applying $2\,|\psi\rangle\,\langle\psi| - I$ is then to make $|\mathbf{x}^*\rangle$ have an amplitude above the mean while all other states have an amplitude below the mean. The desired behavior of the Grover operator is to increase the amplitude of $|\mathbf{x}^*\rangle$ while decreasing the amplitude of the other states. If the Grover operator is applied too many times, this will eventually stop happening. The Grover operator should be applied exactly $\left\lceil \frac{\pi\sqrt{N}}{4} \right\rceil$ times after which a measurement will reveal $\mathbf{x}^*$ with probability close to 1. In the case where $\mathbf{x}$ has two bits, a single application of Grover's operator is sufficient to find $\mathbf{x}^*$ with certainty (in theory). Below is a high level pseudocode for the algorithm.

Algorithm 1 Grover's algorithm

```
Input:
    • An Oracle operator effecting the transformation $|x\rangle|q\rangle \longrightarrow |x\rangle|q \oplus f(x)\rangle$.
Output:
    • The unique bit string $\mathbf{x}^*$ satisfying Eq. (32)
Procedure:
    Step 1. Perform state initialization $|0\ldots0\rangle \longrightarrow |\psi\rangle(\frac{|0\rangle-|1\rangle}{\sqrt{2}})$
    Step 2. Apply Grover operator $\left\lceil \frac{\pi\sqrt{N}}{4} \right\rceil$ times
    Step 3. Perform measurement on all qubit except the ancillary qubit.
```

2.3 Algorithm implemented on IBM's 5-qubit computer

Fig. 7 shows the circuit that was designed to fit the `ibmqx4` quantum computer. The Toffoli gate is not available directly in `ibmqx4` so it has to be constructed from the available set of gates given in Eq. 26.

The circuit consists of state preparation (first two time slots), a Toffoli gate (the next 13 time slots), followed by the $2|\psi\rangle\langle\psi| - I$ operator (7 time slots), and measurement (the final 2 time slots). We use $q[0]$ (in the register notation from Fig. 7) as the ancillary qubit, and $q[1]$ and $q[2]$ as x_1 and x_2 respectively. Note that the quantum computer imposes constraints on the possible source and target of CNOT gates.

Using the simulator, this circuit produces the correct answer $\mathbf{x} = (1,1)$ every time. We executed 1,024 shots using the `ibmqx4` and $\mathbf{x} = (1,1)$ was obtained 662 times with $(0,0)$, $(0,1)$, and $(1,0)$ occurring 119, 101, and 142 times respectively. This indicates that the probability of obtaining the correct answer is approximately 65%. The deviation between the simulator and the quantum computer is due to the inherent errors in `ibmqx4`. This deviation will get worse for circuits of larger size.

We also ran another test using CNOT gates that did not respect the underlying connectivity of the computer. This resulted in a significantly deeper circuit and the results were inferior to the results with the circuit in Fig. 7.

This implementation used a Toffoli gate with a depth of 23 (compared to a depth of 13 here) and obtained the correct answer 48% of the time.

3 BERNSTEIN-VAZIRANI ALGORITHM

3.1 Problem definition and background

Suppose we are given a classical Boolean function, $f : \{0,1\}^n \mapsto \{0,1\}$. It is guaranteed that this function can always be represented in the form, $f_{\mathbf{s}}(\mathbf{x}) = \bigoplus_i s_i x_i \equiv \langle \mathbf{s}, \mathbf{x} \rangle$. Here, $\mathbf{s}$ is an unknown bit string, which we shall call a *hidden string*. Just like in Grover's algorithm we assume that we have a quantum oracle that can compute this function.

The Bernstein-Vazirani (BV) algorithm then finds the hidden string with just a single application of the oracle. The number of times the oracle is applied during an algorithm algorithm is known as