Modified Grover Search for Databases

Quantum Computing Group Project

J. Imbery S. Santamaria M. Signer

2019-07-19

Element Search in Database

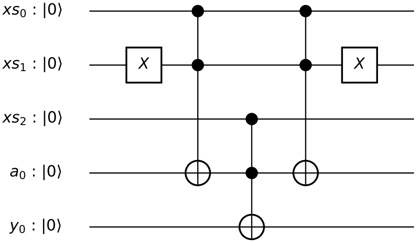
Our $U_\omega : |x\rangle|y\rangle \to |x\rangle|y \oplus f(x)\rangle$

For element search: $f(x) := x = k$

Let $X_{\bar{k}} = \bigotimes_{i=1}^{n} \begin{cases} \mathbf{X} & \text{if the } i\text{-th bit of } k \text{ is } 0 \\ \mathbf{I} & \text{otherwise} \end{cases} : |x\rangle \to \left|x \oplus \bar{k}\right\rangle$, and

$U_\wedge : |x\rangle|y\rangle \to |x\rangle|y \oplus \bigwedge_{i=1}^{n} x_i\rangle$

Then $U_\omega = (X_{\bar{k}})(U_\wedge)(X_{\bar{k}})$



Modified diffusion operator U_s

In the standard Grover search, we mirror around the (fixed) starting state $|s\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{2^n} |i\rangle$: $U_s = \mathbf{I} - 2\,|s\rangle\langle s| = \mathbf{I} - \frac{2}{N}\mathbf{1}$

However, our starting state is given by $|s\rangle = A\,|0\rangle$.

We can use the distributive property:

$$U_s = \mathbf{I} - 2A\,|0\rangle\langle 0|\,A^H = A\mathbf{I}A^H - 2A\,|0\rangle\langle 0|\,A^H = (A)(\mathbf{I} - 2\,|0\rangle\langle 0|)(A^H)$$

So we can mirror about $|s\rangle$ by a isometric transform, mirror about $|0\rangle$, and then transform back!

Iterative search for a single element

Consider the two cases: $k \in S, k \notin S$:

In the first case, this will be a standard Grover search for one of N elements. Then we will get k as a measurement of $|x\rangle$.

In the second case, our Grover iteration will be an identity operation, so in the end we will get a random element from S (which obviously won't be k).

So we can say $k \in S \Leftrightarrow |x\rangle$ measured as k.

Minimum Search in Database

Our $U_\omega : |x\rangle |y\rangle \to |x\rangle |y \oplus f(x)\rangle$

For minimum search: $f(x) := x < k$

This is already implemented in Qiskit aqua with `qiskit.aqua.circuits.FixedValueComparator`

For maximum search: $f(x) := x > k$

`FixedValueComparator` can also implement a $\geq$ comparison. We can turn this into $>$ by adding one to k, or we can use the fact that $\bar{x} = 2^n - 1 - x$, i.e. flipping all bits reverses the order. So we can flip all qubits of x, then compare with $\bar{k}$ and then flip back. The rest of the algorithm for maximum search will be analogous to the minimum version.

Iterative search for the minimum

- *work* $\leftarrow \infty$
- repeat K times:
    - Using Grover search, try to find $x \in S : x < \mathit{work}$
    - *work* $\leftarrow \min(\mathit{work}, x)$

Unlike the single-element Grover search, we don't know $\dim |\mathit{good}\rangle$ (the number of matching elements), so we don't know how many iterations to do. Instead, try different numbers of iterations, up until the number we would need for finding a single element: $1, 2, 4, \ldots, \left[\frac{\pi}{4 \arcsin \frac{1}{\sqrt{N}}} - \frac{1}{2}\right]$. The total time taken will still be $\mathcal{O}(N)$.

It can be proven that at least one of these iteration counts will have a good boosting effect, so it is very likely that we will actually find a smaller element. Because all smaller elements have the same probability of being chosen, the expected number of remaining elements will be cut in half every iteration. As such, we only need $\mathcal{O}(\log N)$ iterations to find the correct minimum.