Quantum Associative Memory (QuAM) implementation in MATLAB

PH5840 - Quantum Computation and Quantum Information - Project Report

Alfred Ajay Aureate R (*Author*)
EE10B052
Dept. of Electrical Eng., IIT Madras,
Chennai, India
alfredajayaureate@gmail.com

Ali Jawed (*Author*)
AE10B053
Dept. of Aerospace Eng., IIT Madras,
Chennai, India
ae10b053@smail.iitm.ac.in

***Abstract*—This project report describes about the implementation of Quantum Associative Memory algorithm by simulating it in MATLAB. Quantum Associative Memory algorithm consists of two quantum based algorithms combined. One is to store a set of patterns, which is converted into binary bit pattern set and then into a state which is the superposition of the qubits that represent those classical binary bit patterns in the set. Now, this state is passed as input to the search algorithm, to find the marked pattern from the set of patterns. In this project, a modified version of the Grover's algorithm is used for this purpose.**

Keywords—quantum computation; quantum algorithms; quantum search; associative memory.

I. Introduction

L. Grover proposed a quantum-based search algorithm for database search in 1996. According to his algorithm, one could search for a particular set of patterns from a set of 2^n patterns, where n is an integer with just $O(\sqrt{N})$ iterations, which is not the case for even the best possible classical search algorithm. This shows the potential of the quantum computational algorithms over the classical ones. But, according to his algorithm, it is required that initially, the quantum state (consisting of superposition of qubits corresponding to different classical bit patterns belonging to the set) should have a uniform probability distribution. Also, we need the complete information of the patterns being searched for. For example, if we need to search for say "0110" from a pattern set, where "0111" is not present, then just searching for the pattern "011" should give us "0110". This is not possible in Grover search, because it has uniform probability distribution, which means, it has non-zero probability for "0111" too, so it would give both the patterns as results. Hence, we have implemented a modified version of the original Grover's search algorithm to overcome this difficulty. So, we use quantum-based approach for this associative memory search. This report shows the improvement by using Quantum Associative Memory (QuAM) that maintains the ability to recall patterns associatively while offering a storage capacity of O(2^n) using only *n* neurons,

First, the classical binary bit patterns, representing any set of patterns that need to be considered for this search, have to be converted to a state, which is the superposition of all qubits which represent those classical bit patterns. This algorithm is called "*Pattern Store*" and the modified Grover's search algorithm used here is called "*Pattern Recall*".

II. Pattern Store Algorithm

A. *Procedure:*

First, we consider the state:

$$|\hat{f}\rangle = |x_1 \dots x_n, g_1 \dots g_{n-1}, c_1 c_2\rangle$$

where *n* is the number of binary bits that represents the pattern. Now, we use three set of qubits *x*(n qubits), *g*(n-1 qubits) and *c*(2 qubits) combined (kronecker product) to form $|\hat{f}\rangle$ for storing the classical bit patterns as qubit patterns. Here, *x* is used to store the final set of qubit patterns; *g* is used to temporarily make sure that we don't have superposition of similar qubit patterns in the *x*. It gives the information to *c* about the similarity. Now, *c* flips the qubits and creates new qubit patterns on *x* by modifying the new x qubit replicas created by $\hat{S}^{p}_{c_1 c_2}$ operator and from the information given by both *g* and *c* itself. By, this procedure, it is also made sure that we get a coherent state superposed with equal probability of all the qubit patterns corresponding to that in the pattern set.

B. *Algorithm:*

Here, the algorithm is described along with an example. Let us consider a pattern set, Z = {'111', '101', '100', '010'} Here we consider m = number of patterns in the set z and n = pattern length / number of bits in each pattern. Also, we assume $\{0\}^n$ to be the (m+1)th element of Z. So, we have '000' as the 5th element of Z, in this example.

Step 1: $Generate\ |\hat{f}\rangle = |x_1 \dots x_n, g_1 \dots g_{n-1}, c_1 c_2\rangle = |0\rangle^{2n+1} = |0\rangle^7 = |000,00,00\rangle$

Step 2: We start from the last pattern, and the variable p runs from p=m to p=1. Here, we start from '010'.

Step 3 to 5: $\hat{F}^{o}_{c_2 x_j}|\hat{f}\rangle$

where, $\hat{F} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ & $\hat{F}^{o} = \begin{bmatrix} \hat{F} & \hat{0} \\ \hat{0} & \hat{I}_2 \end{bmatrix}$

Check which bits are same when z_{pj} is compared with z_{p+1j}, and $\hat{F}^{o}$ is applied on the tensor product of c_2 and only on that jth qubit of *x*. Remember, we have '000' as the 5th element of Z. $\hat{F}^{o}$ is same as CNOT gate, except that it flips the 2nd qubit when 1st qubit is 0. In this example, we thus get, $| 010,00,00\rangle$

Step 6: $\hat{F}^{o}_{c_2 c_1} | \hat{f}\rangle$

Similar to the previous step, the qubit c_1 is flipped if c_2 is 0. Here, we get, $| 010,00,10\rangle$ The above mentioned steps 2 to 6, is together called FLIP.

Step 7: $\hat{S}^{p}_{c_1 c_2} | \hat{f}\rangle$

Now, the $\hat{S}^{p}$ operator acts on the tensor product of the qubits, c_1 and c_2. $\hat{S}^{p}$ is given by,

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \sqrt{p-1/p} & -\sqrt{1/p} \\ 0 & 0 & \sqrt{1/p} & \sqrt{p-1/p} \end{bmatrix}$$

So, we get $\sqrt{\frac{3}{4}} | 010,00,10\rangle + \frac{1}{\sqrt{4}} | 010,00,11\rangle$

Step 8: $\hat{A}^{z_{p_1} z_{p_2}}_{x_1 x_2 g_1} | \hat{f}\rangle$

We consider a three qubit operator $\hat{A}^{ij}_{pqr} =$ 3 – qubit operator that acts on the tensor product of the qubits *p*, *q* and *r*. It actually flips the qubit *r*, if *pq=ij*. So, we have $\hat{A}^{00}, \hat{A}^{01}, \hat{A}^{10}$ *and* $\hat{A}^{11}$.

$for\ example$: $\hat{A}^{00} = \begin{bmatrix} \hat{F} & \hat{0} \\ \hat{0} & \hat{I}_6 \end{bmatrix}$

In our algorithm, we use $\hat{A}^{z_{p_1} z_{p_2}}_{x_1 x_2 g_1}$, it acts on the tensor product of x_1, x_2 *and* g_1. It flips the qubit g_1, if $x_1 x_2 = z_{p_1} z_{p_2}$, where z_{p_1} *and* z_{p_2} are p1th and p2th bit of the pth pattern respectively, from the pattern set. In this example, we get, $\sqrt{\frac{3}{4}}$ $| 010,10,10\rangle + \frac{1}{\sqrt{4}} | 010,10,11\rangle$

Step 9 to 10: $\hat{A}^{z_{p_k} 1}_{x_k g_{k-2} g_{k-1}} | \hat{f}\rangle$

Similar to the previous step, we now apply it on the tensor product of x_k, g_{k-2} *and* g_{k-1} qubits. So, g_{k-1} is flipped, if z_{p_k} is x_k and g_{k-2}=1. This, is repeated for incremental values of k=3 to k=n. We, get, $\sqrt{\frac{3}{4}} | 010,11,10\rangle + \frac{1}{\sqrt{4}} | 010,11,11\rangle$

Note that here the qubit *g* checks the similarity of the *x* qubits and becomes 1 as long as the *x* qubits match, starting from the 1st qubit of *x*. Here, since both *x* matches for all the qubits, starting from the first qubit, we get *g* to be '11'. *g* would always be ones followed by zeros.

Step 11: $\hat{F}^{1}_{g_{n-1} c_1} | \hat{f}\rangle$

Again we use, 2-qubit operator, but this $\hat{F}^{1}$ is the normal CNOT gate, which flips the 2nd qubit when 1st qubit is 1. Here, it is applied on the tensor product of g_{n-1} *and* c_1. Thus, we get, $\sqrt{\frac{3}{4}} | 010,11,00\rangle + \frac{1}{\sqrt{4}} | 010,11,01\rangle$

Note that, c_1, gets flipped here, only when we have all the qubits of *x* being similar, like in this example.

Step 12 to 13: $\hat{A}^{z_{pk} z_1}_{x_1 g_{k-2} g_{k-1}} | \hat{f}\rangle$

This step is similar to the steps 9 to 10, except for the fact that here we are starting from the values of *k=n* to *k=3*. In this example, we get $\sqrt{\frac{3}{4}} | 010,10,00\rangle + \frac{1}{\sqrt{4}} | 010,10,01\rangle$

Step 14: $\hat{A}^{z_{p_1} z_{p_2}}_{x_1 x_2 g_1} | \hat{f}\rangle$

This final step is again similar to the step 8.

Actually, since the function of *g* qubit is over, steps 12 to 14 is used to again flip them back all to 0's. So, after one full iteration of all 14 steps for the first pattern, we get

$\sqrt{\frac{3}{4}} | 010,00,00\rangle + \frac{1}{\sqrt{4}} | 010,00,01\rangle$

The above mentioned steps 8 to 14, is together called SAVE. Note, in the above cases, we could see that x qubits are same in both of them. So, if we try to plot the superposed vector, $| phi\rangle$, we would get probabilities corresponding to the qubit '010' to be $\sqrt{\frac{3}{4}} + \frac{1}{\sqrt{4}}$ = 1.366>1. But, it does not affect the final results. Now, again all the steps 2 to 14 are repeated for all the other patterns. Note that, when $c_1 c_2 = 01$, the corresponding *x* qubit and its probability remains unchanged for the rest of the algorithm. Thus, after iterating over all the patterns, we get,

$\frac{1}{\sqrt{4}} | 010,00,01\rangle + \frac{1}{\sqrt{4}} | 100,00,01\rangle + \frac{1}{\sqrt{4}} | 101,00,01\rangle + \frac{1}{\sqrt{4}} | 111,00,01\rangle$

Here, the qubits *g* and *c* are discarded, and only *x* qubit is used to represent the required pattern set. Thus, at the end of Pattern Store algorithm, we get,

$$\frac{1}{\sqrt{4}} | 010\rangle + \frac{1}{\sqrt{4}} | 100\rangle + \frac{1}{\sqrt{4}} | 101\rangle + \frac{1}{\sqrt{4}} | 111\rangle$$

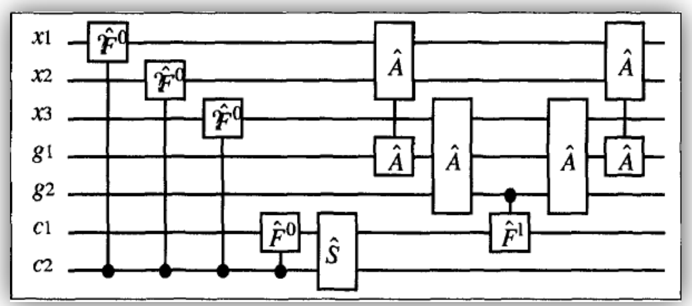

Figure 1: Quantum circuit for the Pattern Store for the example

III. Pattern Recall Algorithm (Modified Grover's Search Algorithm)

A. *Normal Grover Search Algorithm:*

Step 1: In the Grover search algorithm, we start with the state $|0\rangle^{n} = |0\rangle^{3}$.

Step 2: We then apply $H^{\otimes n}$ to make it now uniformly distributed over all qubits.

Step 3: Then, we apply the oracle I, that flips the coefficients of the qubits which are marked.

Step 4: After that, the oracle G is applied over all the n qubits. It is actually, the combination of $H^{\otimes n}$, I_0 and $H^{\otimes n}$, where I_0 flips the coefficients of the qubits that are not $|0\rangle$. Whereas, the G operator flips the qubits with respect to the average of the probabilities associated with each qubit of $|phi\rangle$. It is usually called the inversion about the mean. The above mentioned last two steps (steps 3 and 4), applying oracle I and applying the oracle G, is iterated for $\left\lceil \frac{\pi}{4}\sqrt{N} \right\rceil$ times to get the marked patterns.

B. *Modified Grover's search algorithm:*

Step 1: In this modified version of the Grover search algorithm, we start with the state $|phi\rangle$ (where it is the superposition of some m qubits, each with probability $\sqrt{\frac{1}{m}}$ and m need not be an integer power of 2) and apply the oracle I, that flips the coefficients of the qubits which are marked.

Step 2: After that, the oracle G is applied over all the n qubits, as in the normal algorithm.

Step 3: Then, we apply the oracle I_p, which flips the qubits that belonged to the initial state $|phi\rangle$.

Step 4: After that, again, the oracle G is applied over all the n qubits.

Step 5: Then, we apply the oracle I, that flips the coefficients of the qubits which are marked.

Step 6: Finally, after that, again, the oracle G is applied over all the n qubits.

The above mentioned last two steps (steps 5 and 6), applying oracle I and applying the oracle G, is iterated for $\left\lceil \frac{\pi}{4}\sqrt{N} - 2 \right\rceil$ times to get the marked patterns.

IV. Comments

In this project, the simulation is done for cases upto, n=8 and upto m=256, whereas the code does not have any limitations on n, nor m. The associative memory part is also simulated, the code takes in any number of bits as input and searches from the set. It also considers any number of marked items. For the associative memory application, having more than one marked item explicitly is avoided here, as it will anyway have many marked items, when just fewer than n qubits alone are mentioned for searching.

References

[1] Ventura, D., Martinez, T.: Quantum associative memory. Inf. Sci. **124**(1), 273–296 (2000)

[2] D. Ventura, T. Martinez, Initializing the amplitude distribution of a quantum state, Foundations of Physics letters 12 (1999) 547-559.

[3] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*, (Cambridge University Press, 2000)

[4] P. Kaye, R. Laflamme and M. Mosca, *An Introduction to Quantum Computing*, (Oxford University Press, 2007)

[5] Grover, L., "A Fast Quantum Mechanical Algorithm for Database Search", Proceedings of the 28th Annual ACM Symposium on the Theory of Computing, ACM, New York, pp. 212-19, 1996.