Quantum Research Project

Pierriccardo Olivieri

March 2020

Abstract

This project aims to analyze in a computer science perspective the quantum version of random walks: the quantum walks. This model of computation can be seen as a building block to construct new algorithms, here we are interested in a graph application in the domain of discrete time quantum walks. The question that this research want to answer is if a speedup w.r.t. classical is possible and discuss eventual limitations.

Contents

1 Introduction

1.1 Random Walks

A random walk, also known as a stochastic or random process, is a mathematical object which describe a path constitued by random step over a mathematical space. A common example is a random walk in a line, consider the integer set Z, starting from a certain point, for instance x = 0, the path is defined by randomly choose to move left or right, the movement is achieved increasing or decreasing the value of x by 1. Tossing a coin will help in choosing randomly, with equal probability, the next step to take. This example could be generalized by increasing the dimension of the mathematical space considered, in a cartesian plane the starting point will have two coordinates and the possible moves becomes 4, and there are many other possible generalizations.

The random walk can be divided in two major classes, discrete-time and continous-time random walk. Intuitively as the names suggest the difference between this two class is in the time function that could be integer or real, for a more formal definition consider the random walk as a system composed by a family of random variables X_t the variables X_t will measure the system at time t, now if we consider $t \in \mathbb{N}$ is a discrete-time stochastic process, otherwise if $t \in \mathbb{R}^+ \bigcup 0$ is a continous-time stochastic process.

Here we focus on the discrete-time, in par-

ticular we bind this to graphs considering the random walk on a cyclic graph of order N.

1.2 Quantum Walks

The Quantum version of the random walks is called Quantum Walks, also here there is a disctinction between the two model of discrete quantum walks and continous quantum walks, the focus of this research remain in the discrete-time domain also for the quantum side.

The discrete quantum walk described here is called coined Discrete Quantum Walk on a Line (Coined DQWL), there are also versions without coin. It's worth introducing the Coined QDWL in a formal and generic way, then apply to a more concrete example. A formal description starts considering the three main components of a Coined QDWL: A walker operator, a coin, an evolution operator usually called shift operator.

The Walker represents the position of our system and is defined in a Hilbert space inifite but countable $\mathcal{H}_p$, a vector in that space represents the postion of the walker, $|position\rangle \in \mathcal{H}_p$

The coin operator is defined in a two-dimension Hilbert space, if we consider as basis state $|0\rangle$ and $|1\rangle$ then the coin space will becomes $\mathcal{H}_c = |0\rangle, |1\rangle$ and with $|coin\rangle \in \mathcal{H}_c$.

The complete system finally will be in a Hilbert space composed by the Kronecker product of the two spaces above defined $\mathcal{H} = \mathcal{H}_p \otimes \mathcal{H}_c$. A state of the coined DQWL can be defined by the vector:

$$|\phi_{initial}\rangle = |position\rangle_{initial} \otimes |coin\rangle_{initial} \quad (1)$$

The evolution operator, also called shift operator will actually perform a step starting from the initial position, apply this operator to the system is equal to toss a coin and depending on the outcome move the walker to the left or to the right, we can do this by increment or decrement the actual position by 1. A possible form of the shift operator could be described by this formula:

$$S = |0\rangle_c \langle 0| \otimes \sum i \, |i+1\rangle_p \langle i| + |1\rangle_c \langle 1| \otimes \sum i \, |i-1\rangle_p \langle i| \quad (2)$$

Finally we can see a walker as operator itself, called U and given by:

$$U = S \times (C \otimes I_p) \quad (3)$$

Applying this operator to a given system is equal to perform a step of a random walk. A more formal explanation can be found in [Ven08] and [Kem03]. The example on the following section apply this general concept to a cyclic graph.

2 Coined Quantum Walk

The following example shows how to implement a Coined discrete quantum walk on a cyclic graph with N = 8 nodes. This can be achieved using the coined DQW on a line where the line is the cylic graph. First we need to encode the graph nodes in a binary notation to fit with qubits. In general a graph with 2^n nodes needs n encoding bit, in our case since we have 2^3 nodes we can use 3 bit to encode. The Figure 1 below shows how we can bind the qubits to the nodes of the graph.

2.1 Components description

We can fit now the 3 main components mentioned in the previous section for this specific example.

Starting from the **walker operator**: for this version we need to encode the nodes of the graph in 3 qubits, usually we need log(N) qubits, in general this is not always true, we will see why in the next section. Using 3 qubits

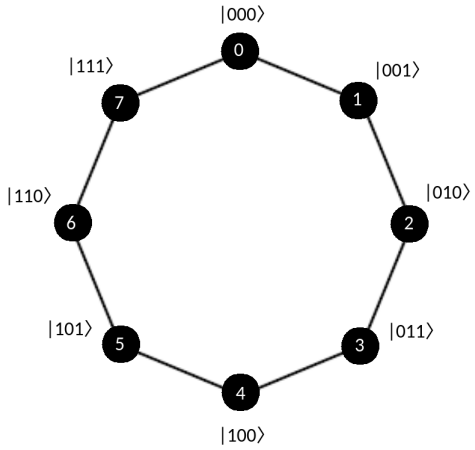


Figure 1: Cyclic graph with 8 nodes, and the respective position state in qubits version

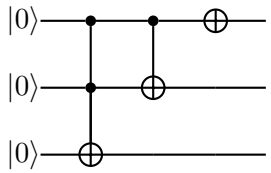
the postion of the walker is given by the binary encoding of the node, for instance the node 0 is represented as $|000\rangle$.

For the **coin operator** operator, we will use an Hadamard coin, that consists in apply the Hadamard operator to the system.

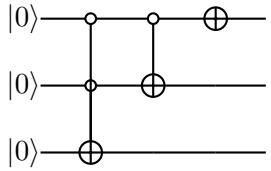
The **shift operator** operator in this case will move the actual position that we call $|i\rangle$ to one of the adjiacent nodes, that corresponds to $|i+1\rangle$ or $|i-1\rangle$, which is decided by the outcome of the coin operator.

2.2 Circuit for the CQW

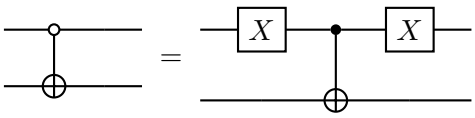
To implement the circuit in qiskit we need to translate the operator defined in quantum circuits, first we need 1 qubit for the coin and 3 for the position. The Hadamard can be easily implemented using an Hadamard gate on the first qubit. Then we need a circuit to perform an increment and a decrement on the initial state, this is less trivial and to achieve that we need two sub circuit that uses multi controlled toffoli gates. The circuit below represents an incrementer circuit for 3 qubits.



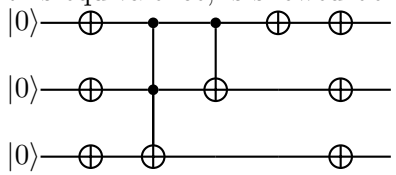
The decrement circuit is similar but uses negative controlled not gates, the circuit below shows the decrement circuit for 3 qubits.



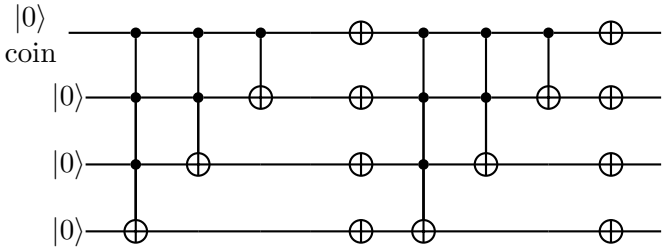
The negative controlled not can be represented by negate before and after the controlled not, the equivalence circuit below clarify this explanation, for a detalied explanation look at [NC10].



Finally the decrement circuit defined above, using this equivalence, is showed below.



A similar implementation in detail is covered in [DW07]. The final circuit by combining all the components defined is showed below.



It's worth to make some comments about it, this circuit represents the U operator, in fact the Hadamard coin will randomly choose the direction to take and activate the increment or decrement sub circuit. Therefore this corresponds to a single iteration of the walk, by successively apply this circuit we can perform a random walk on the cyclic graph. The code for this circuit can be found in the Appendix A at the end of this document.

3 Results, Generalizations and Limitations

3.1 Results

The circuit obtained in the previous section can be applied to a system o 4 qubits several times to perform a random walk. Fig. 2 shows the results of a 100 steps random walks, obtained by apply the circuit 100 times.

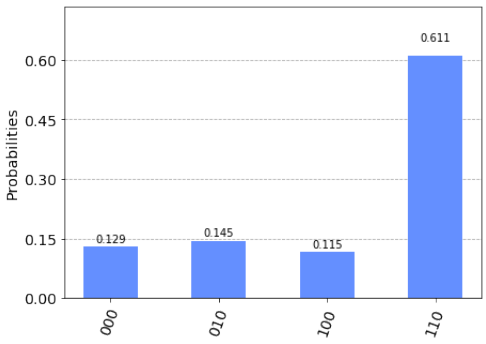


Figure 2: results of a 100 step random walk

As we can notice in the picture the probailities of find the walker in a given position are quite asymmetrical and also odd numbers have zero probability of being measured, this is due to the Hadamard coin, in fact, differently from classical, we can see a true randomness behavior different from a Gaussian distribution that we will obaserve in a classical random walk. This asymmetrical behavior can be modified changing the initial state or changing the coin, a more formal description of this behavior is showed in [Ven08].

3.2 Generalizations

The example presented is just an application to cyclic graph, also the 3 main component presented, the walker, the coin and the shift operator are quite generic. This method in fact can fit different type of problems and graph. Starting from the coin there are many others operator that we can use, which are symmetrical, other examples are the Groover coin or a Balanced coins that get rid of this asymmetrical behavior of the Hadamard coin. For more details about this coin an introductory summary can be found in [Kem03], but there are also Quantum walks without the coin, as it is mentioned in [Ven08].

The interesting thing is that the example presented can be adpted to a variety of other graphs, for instance we can build a circuit able to perform a quantum walks also for completed graphs, hypercube, glued trees and others. Obviusly, changing the graph, the circuit needs to be adapted but the idea remains the same. A very usefull references that shows circuits for the type of graphs mentioned is [DW07]

An important remark concern the total qubits needed, as mentioned in the previous section if the number of qubits is greather than 2 we need to use multi toffoli gate, this gates can be obtained by combining several toffoli gates, with the help of some ancillary qubits, those ancillary qubits are needed just to construct the multi controlled toffoli. In Fig. 3 below is showed how to implement a multi controlled toffoli gates with 5 controls qubits, this image is taken from [NC10]. In general following this procedure we need a number of ancillary equals to controls qubits - 1.

The number of ancillary qubits inpacts also in the efficiency of the circuit. In the next sec-

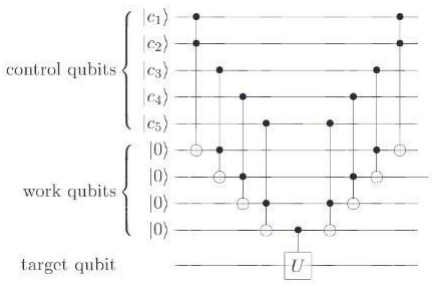


Figure 3: Practical representation of multi toffoli gate

tion will be provided a more formal description of efficiency.

3.3 Limitations

The method presented, as we just said, can be generalized to other types of graphs but unfortunately is limited to undirected graphs without weights, since various application require weighted and/or directed graphs we need something more generic. In the next chapter is showed a method that can achieve quantum walks also for undirected and weighted graphs. Before presenting this method, we need a brief introduction to Markow Chain, using the Markow Chain representation of the graph we can reduce the limitations and work with directed and weighted graphs.

4 Szegedy quantum walk introduction

The following method proposed by Szegedy helps to reduce some limitations discussed in the previous section. In order to talk about Szegedy algorithm we need to define introduct some concepts.

4.1 Markow chains

A Markow chain is a stochastic process that consists in a sequence of random variables X_n with $n \in \mathbb{Z}^+$ such that $P(X_n|X_{n-1}, X_{n-2}, ..., X_{n-N}) = P(X_n|X_{n-1})$. If is time-independent can be represented by a matrix P called transition matrix. Such that the sum of each row of P is equal to 1.

4.2 application to graphs

We can use then a Markow Chain to perform a random walk on a graph. First, considering a graph G(V,E) we construct the adjacency matrix as follows:

$$A_{i,j} = \begin{cases} 1 if (v_i, v_j) \in E \\ 0 otherwise \end{cases} \tag{4}$$

References

[Kem03] J Kempe. "Quantum random walks: An introductory overview". In: *Contemporary Physics* 44.4 (July 2003), pp. 307–327. ISSN: 1366-5812. DOI: 10.1080/00107151031000110776. URL: http://dx.doi.org/10.1080/00107151031000110776.

[DW07] B. L. Douglas and J. B. Wang. *Efficient quantum circuit implementation of quantum walks*. 2007. arXiv: 0706.0304 [quant-ph].

[Ven08] S. Venegas-Andraca. *Quantum Walks for Computer Scientists*. 2008.

[NC10] Michael A. Nielsen and Isaac L. Chuang. "Quantum circuits". In: *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University

Press, 2010, pp. 171–215. DOI: 10.
1017/CBO9780511976667.008.

A Coined Quantum walk

This code below is referred to coined quantum walk, using Hadamard coin for a cycle graph with number of nodes N=3.

```
from qiskit import *

#increment operator for a 3-bit state register
def increment_op(circuit):
    qr = circuit.qubits
    circuit.mct([qr[0],qr[1],qr[2]],qr[3],None,mode='noancilla')
    circuit.ccx(qr[0],qr[1],qr[2])
    circuit.cx(qr[0],qr[1])
    circuit.barrier()
    return circuit

#decrement operator for a 3-bit state register
def decrement_op(circuit):
    qr = circuit.qubits
    circuit.x(qr)
    circuit.barrier()
    circuit.mct([qr[0],qr[1],qr[2]],qr[3],None,mode='noancilla')
    circuit.ccx(qr[0],qr[1],qr[2])
    circuit.cx(qr[0],qr[1])
    circuit.barrier()
    circuit.x(qr)
    return circuit

#construct the circuit for one step of the random walk
def random_walk_step(circuit):
    #increment operator circuit
    qr_incr = QuantumRegister(4)
    increment_circ = QuantumCircuit(qr_incr, name='increment')
    increment_op(increment_circ)
    increment_inst = increment_circ.to_instruction()

    #decrement operator circuit
    qr_decr = QuantumRegister(4)
    decrement_circ = QuantumCircuit(qr_decr, name='decrement')
    decrement_op(decrement_circ)
    decrement_inst = decrement_circ.to_instruction()

    circuit.h(qr[0])
    circuit.append(increment_inst, qr[0:4])
    circuit.append(decrement_inst, qr[0:4])

    return circuit

def random_walk(steps, circuit):
    for i in range(0, steps):
        random_walk_step(circuit)
    return circuit
```