Quantum Circuit Optimilization Using Monte Carlo Tree Search and Reinforcement Learning

Author: Liza Darwesh
BSc. Thesis

Supervisors:
Ariana Torres, SURF
Damian Podareanu, SURF
Dick Heinhuis, HvA

Hogeschool van Amsterdam

A Preliminary Bachelor's Thesis
Graduation Assignment for SURF BV
Department of Innovation
Amsterdam University of Applied Science

April 19, 2022

Chapter 1

Introduction

1.1 Motivation

It is fascinating to think about the fact that the computing power of a military computer from 50 years ago, that was the size of an entire room, are in today's regular computers. However, even with the phenomenal growth we made in technology, there remain problems that regular computers just can't solve.

Quantum computers are expected to make a breakthrough in chemical and biological engineering through the discovery and manipulation of molecules; encryption for cyber security; the processing of very large quantities to aid in artificial intelligence; and the pricing of complex assets in finance [1]. The quantum computers from nowadays may be able to perform tasks which surpass the capabilities of today's regular computers, but the world is not there yet.

1.2 Problem

One of the barriers of programming a quantum computer is the loss of information in a qubit. Think of a spinning coin with information: either heads of tails. Eventually the coin will stop spinning and land on one of the sides. This is the same for a quantum system. The algorithms that are written for a quantum computer contain noise, which can ensure that the information provided is unreliable. The way algorithms can be operated is by writing them into quantum gates. They are the same as the regular logic gates, but with quantum phenomena, which will be explained later in the paper. So basically, mathematical algorithms are transformed into circuits of operations. These circuits are written in Quantum Assembly, a language that the quantum computer understands. Each circuit has a defined number of logical quantum bits (qubits). These qubits are what the gates are interacting with. In the moment of processing, the logical qubits in the circuit will then be mapped into the topology of the quantum computer, which is a connectivity architecture of the qubits inside the quantum computer, also refereed as physical qubits. The problem here lies in the connectivity matching. When programming a circuit, one must satisfy the topology. For instance, say two qubits need to interact with each other through a gate. If these qubits are not connected to each other, this operation will be inoperable, which means the whole circuit

cannot be executed. So there needs to be a way to make circuits operable while satisfying the connectivity constraints of the quantum computer. A way to do this is by adding a so called SWAP-gate in the circuit, that causes the qubits to flip. This flip ensures that the qubits come closer to each other or even become connected. The problem with the swap gate is that it is not a known logic gate for the quantum computer. The swap gate is performed by placing 3 CNOT gates one behind the other. This ensures that the circuit becomes larger in depth and therefore also more sensitive to decoherence. So there needs to be an optimization in placing these SWAP gates to ensure that the information is reliable.

1.3 Organisation

The research of this thesis is done for the organisation SURF. A cooperative association of Dutch educational and research institutions. Universities, universities of applied sciences, MBO institutions, UMCs and research institutes work together within SURF to purchase or develop the best possible digital services. The company owns supercomputers which are available to complete high performance processes. Within the company there are multiple departments including quantum innovation, machine learning, and high performance computing. Which will be useful for this research.

1.4 State-of-the-art

There are already a number of studies done in this area, for example a state-of-the-art research was the one of Pozzi, et al. [2], where he used Reinforcement Learning and something called Quantum Annealing to optimize the number of SWAP-gates in a circuit. His research was based on a random logical qubit allocation on the physical qubits. When creating an initial qubit placement, the number of SWAP-gates can be reduced. Another recently published study is done by Sinha, et al [3], where they managed to minimize the number of SWAP gates for a random qubit allocation using Monte Carlo Tree Search in combination with Reinforcement Learning.

1.5 Research

Both papers discussed the need to look into initial qubit placement, which will also ensure that the SWAP gates are minimized in addition to using Reinforcement Learning. With this in mind, research will be conducted on quantum circuit optimization using the Monte Carlo Tree Search in combination with Reinforcement Learning. An initial qubit placement will be used for this.

This paper answers the main question: 'How can quantum circuit routing be optimized using Reinforcement Learning applied on Monte Carlo Tree Search?' This question will be answered by means of the following sub-questions:

- What is quantum circuit routing?

- What has already been done to successfully accomplish quantum circuit routing?
- How can the initial qubit placement procedure be performed?
- How does the Monte Carlo Tree Search work on optimizing the number of SWAP-gates?

This research paper is organised as follows. Section ?? offers a description of basic knowledge about quantum computing phenomenon and techniques needed for understanding the main question. Section ?? demonstrates what research has already been done by others. Section ?? describes the sub-problem initial qubit placement and how to solve it. Section ?? provides an overview of the experimental setup, which is how the Monte Carlo Tree Search is put together with the Reinforcement Learning. Section ?? shows the results of what the reinforcement learning model delivered. Finally, in Section ?? and ?? states the discussion about the research that is performed and the final conclusion.

Chapter 2

Quantum Circuit Routing

It is recognized that quantum computers can provide revolutionary developments. This chapter explains what a quantum computer is and how it can be programmed. It also explains in detail what the problem is that will be optimized.

2.1 The Qubit

A quantum computer can be seen as a regular computer that stores information and performs operations using quantum mechanics. A regular computer stores all its information such as numbers, text, and images, in series of 0's and 1's. These units are called bits. The way a quantum computer stores information is through the use of quantum bits (qubits). The difference between a bit and a qubit is that a qubit can carry not only the information 0 or 1, but also 0 and 1 on top of each other. This phenomenon is called superposition. A well-known example of superposition is the Schrödinger's cat example, where a hypothetical cat is illustrated in a closed box with a bottle of poison. It is not certain whether the cat is alive or dead unless it is checked. Before checking, the cat finds itself in a situation where it has a 50% chance of being dead or alive. This is the same with a qubit, there is no certainty whether it will hold 0 or 1, only a 50% probability that it will be one of the two. An illustration of a bit and a qubit can be find in figure 2.1.

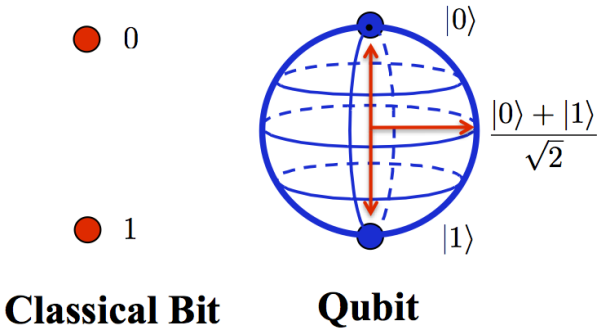

Figure 2.1: Regular bit and quantum bit illustration

2.2 Quantum Circuits

A regular computer operates by means of directions (algorithm) that are provided in the form of a script. This script then tells the computer processor what calculations have to be done to arrive at a certain output. These scripts can be written in different programming languages and the computer translates this into binary language, to control the logic gates inside the processor. The logic gates provide the calculations necessary for the operations. This principle, of programming gates that do calculations to perform a certain operation, works the same in a quantum computer. A script is written that determines which quantum gates should operate. An example of a gate that will be discussed a lot in this thesis is the Controlled NOT gate (CNOT-gate), see figure 2.2.

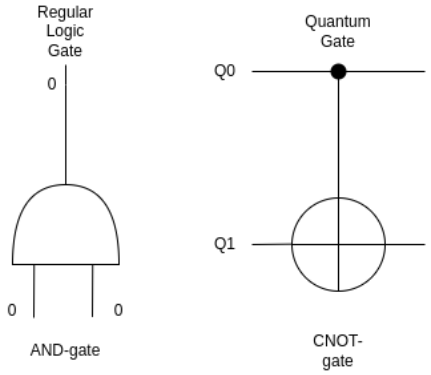

Figure 2.2: Example of a logic gate in a regular computer (left), example of a Quantum gate (right)

This script then creates a circuit consisting of the sequence of the quantum gates that were defined in the script. The algorithm that has been programmed is thus represented as a circuit with quantum gates. An example of a circuit can be seen in figure 2.3. The circuit illustrates between which qubits operations have to be performed. The circuit of figure 2.3 illustrates a four qubit circuit consisting of CNOT-gates with each interacting with two qubits. Usually, qubits are represented in Dirac notation [4], but in this thesis that is not relevant and the qubits will be represented with a capital Q.

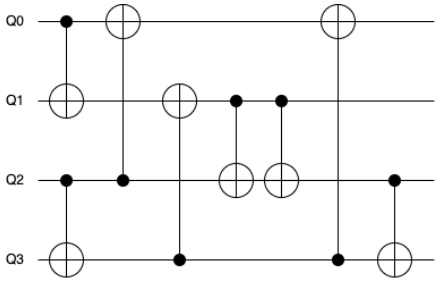

Figure 2.3: A quantum circuit extracted from a programmed script

Each horizontal line in the circuit indicates the 'path' of a qubit, each path can have a quantum gate through which the qubit has to pass. Ultimately, the paths can be read to obtain a final result. As mentioned earlier in the example of

Schrödinger's cat, there is a 50% chance that the cat is alive or dead unless the box is put up for checking. Reading or measuring a circuit forces the qubits to choose a state, either 0 or 1. The moment it chooses one of the two, the qubit collapses to a regular bit with the one value. This means that the output of the circuit will represent an ordinary series of bits, just like in a regular computer.

2.3 Topology

The step after feeding a quantum circuit to the quantum computer is the process in which the defined qubits in the circuit (logical qubits) are allocated to the qubits in the hardware (physical qubits). The logical qubits are, as it were, assigned a location on the hardware. The form in which these qubits can be allocated depends on the connectivity architecture of the physical qubits. This means that the qubits in the hardware are attached to each other in a certain form. This architecture is also known as the topology of the quantum computer.

There are different types of topologies. For example, figure 2.4b shows the topology of Rigetti's Aspen-4 quantum computer with a linear structure, or figure 2.4e shows the topology of Google's Sycamore, which consists of more connected qubits in a grid structure.

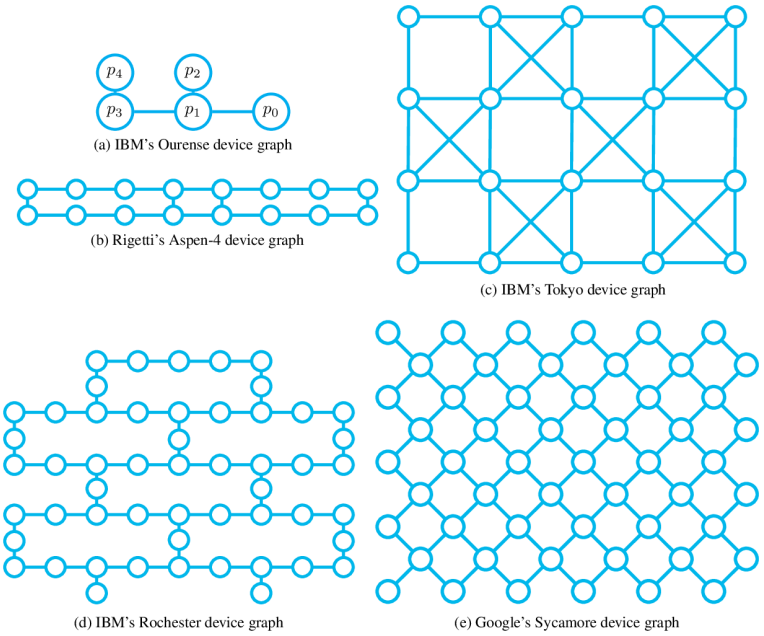

Figure 2.4: Quantum topologies of big tech companies' quantum computers. Source: [5]

These connectivities determine which qubits are allowed to interact with each other. Imagine the topology of figure 2.5 and the circuit of figure 2.3.

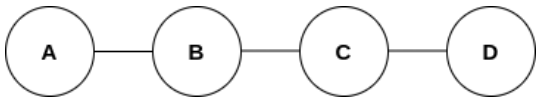

Figure 2.5: Example of a linear topology.

There are several ways to map the circuit's logical qubits into the computer's physical qubits, see figure 2.6. It can be done in sequence, randomly, but also with an initial placement, where the allocation is based on the qubits that need to interact in the circuit.

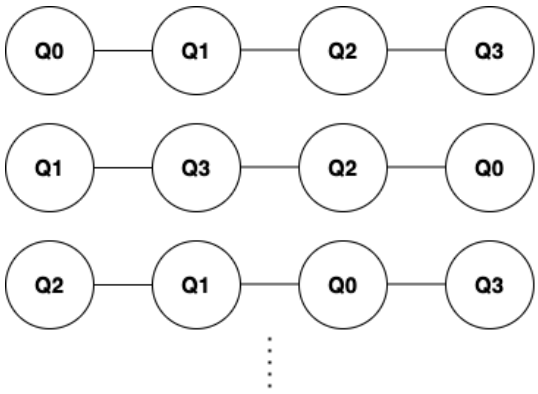

Figure 2.6: Different combinations of qubit mapping.

After mapping the logical qubits, the operations in the circuit will be performed. The moment interactions have to be performed in the circuit between qubits that have no connection in the topology, the operation cannot be performed, after which the entire circuit is rejected by the quantum computer. This appears to be a common problem when writing quantum algorithms. One way to make the circuit perform on a quantum computer is to add SWAP-gates to the circuit 2.7.

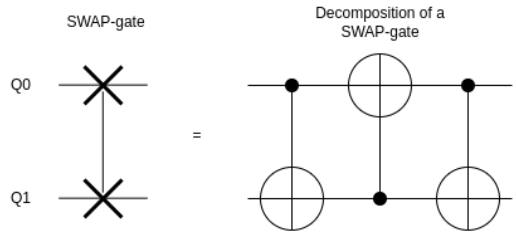

Figure 2.7: A SWAP-gate and its decomposition.

The SWAP gate will ensure that the qubits, that interact with the SWAP-gate, will switch positions in the quantum computer. This means that the qubits that have to operate with each other have a connection in the topology. This can be done over the entire circuit, until all operations are executable hardware-based.

A SWAP-gate is not a standard gate known to the quantum computer. A SWAP-gate can be simulated by placing three CNOT gates in a row, as shown in figure 2.7. Because there are actually three extra gates added per SWAP gate, this will ensure that the circuit will have a greater depth.

2.4 Decoherence

Qubits can be compared to a spinning coin. Eventually the coin will stop spinning and land on heads or tails. A qubit is also spinning between 0 and 1, but will also

eventually stop spinning, after which it will end up at 0 or 1. This phenomenon of breaking out of the superposition, after which it has to choose a state is called decoherence. The lifespan of a qubit differs per hardware composition of the quantum computer. Based on the temperature and network in the quantum chip, it can be ensured that the lifespan of a qubit is longer. The longest achieved so far by a qubit is 50 seconds, but the most common is somewhere between 15-120µ seconds [6].

Because qubits decohere, it causes parts of information to be lost. As a result, the output information will be unreliable. For this reason, it is important to set up a circuit that is small in depth, so that it can be quickly executed by the quantum computer while the qubits last. This is a tricky problem when a large algorithm has to be executed, where SWAP-gates are essential.

Chapter 3

Previously researched work

This chapter will provide a picture of what has been done before to optimize circuit routing and what has been recommended by these researchers for further research.

Chapter 4

Monte Carlo Tree Search with Reinforcement Learning

This chapter will describe how the Monte Carlo Tree Search (MCTS) in combination with Reinforcement Learning (RL) has been applied to optimize quantum circuit routing.

4.1 Implementation of the MCTS

The idea of using the Monte Carlo Tree Search (MCTS) comes from AlphaGo Zero [7], where using MCTS, a neural network is trained to take actions in the Chinese game of Go. MCTS uses exploration and explotation simulate a whole tree of actions, from which every node in the tree has a success probability.

Bibliography

[1] F. Bova, A. Goldfarb, and R. Melko, "Quantum computing is coming. what can it do?" Jul 2021. [Online]. Available: https://hbr.org/2021/07/quantum-computing-is-coming-what-can-it-do 1.1

[2] M. G. Pozzi, S. J. Herbert, A. Sengupta, and R. D. Mullins, "Using reinforcement learning to perform qubit routing in quantum compilers," Jul 2020. [Online]. Available: https://arxiv.org/abs/2007.15957 1.4

[3] A. Sinha, U. Azad, and H. Singh, "Qubit routing using graph neural network aided monte carlo tree search," Mar 2022. [Online]. Available: https://arxiv.org/abs/2104.01992 1.4

[4] C. Gronlund, M. Li, and S. Lopez, "Dirac-notatie - azure quantum," Apr 2022. [Online]. Available: https://docs.microsoft.com/nl-nl/azure/quantum/concepts-dirac-notation 2.2

[5] B. Tan and J. Cong, "Optimality study of existing quantum computing layout synthesis tools," IEEE Transactions on Computers, vol. 70, pp. 1363–1373, 2021. 2.4

[6] D. Shaw, "Quantum hardware - into the quantum jungle," Jul 2020. [Online]. Available: https://www.factbasedinsight.com/quantum-hardware-into-the-quantum-jungle/ 2.4

[7] D. Silver, "Alphago zero: Starting from scratch," Oct 2017. [Online]. Available: https://www.deepmind.com/blog/alphago-zero-starting-from-scratch 4.1