Qilimanjaro Hackathon- Team name: Disqualified

Members: Luca, Rens, Vasudha, Giovanni

1. Introduction
2. Software Challenge
   1. Challenge –1
   2. Challenge –2
   3. Encoding the qubits
   4. Writing the Cost function and Constraints
   5. Writing the mixer matrix and Possible states
   6. Conceiving the hardware implementation
3. Hardware Challenge
   1. Find a totally non-trivial optimization problem in [11] that you can implement with 3-5 qubits.
   2. Explaining the plots from the PW SW method.
4. Introduction:

This hackathon was excellent in introducing us to QAOA, AQO and also in understanding how annealing is done using hardware(simulations).

2. Software Challenge:

The traveling salesman problem

We were challenged to solve the traveling salesman problem with the use of digital quantum computing with 5 nodes and a limited number of available q-bits, namely 6. Normally this scales with N^2 q-bits needed, in which N is the number of nodes within the system, or otherwise known as the number of camps traveled through. For our case, we will look at the situation of climbing the Mount Kilimanjaro in light of the challenge being given by Qilimanjaro. However, we will look into generalising this for all possibilities with 5 nodes, such that we can solve it for all possible routes that can be added/subtracted from this case. The case of the Mount Kilimanjaro is given in the below figure:

In this, each path has a certain weight w\_{ij}, such that the time it takes to travel from point I to point j is this weight. We want to minimise the amount of time it takes to go from our base camp, Katanga Camp, to our final camp, Kibo. Thus we can say we have already constrained our system such that Katanga camp, noted as node 0, and Kibo, noted as node 4, already are defined. This decreases the number of degrees of freedom within our system already. The most important is to encode our system in such a way that we only have to use 6 q-bits.

To do this, we first looked at a simpler problem of 3 nodes, in which we used 3 q-bits. We have defined it in a similar way as the 5 node system, which results in the structure depicted in the below figure:

This is again with the weights defined in the same way. We solved this by first defining a Hamiltonian to mix the states into a superposition of all feasible states, which is done by just using operators that change certain 0’s to 1’s. This is then evaluated with a cost function as depicted below:

After solving the problem for 3 nodes, we started to figure out how to do it for 5 nodes, with the constraints of using 6 q-bits. We used a multitude of methods in our design project, as depicted in the below figure:

In this we note that edges can be used for 5 nodes with 6 q-bits. We tried to encode our states with binary encoding, but the coupling to the different weights used too many q-bit gates.

We also tried to use jumps, which was a way to say how far we needed to jump to select the next place we jump to. Our most original idea, that was absolutely horrible was using radiuses. This would mean that adding up the radiuses of both nodes from where you came, to where you were going adds up to the weight of the path. This was however not doable.

Then we tried to apply QUBO, but it seemed like it was not possible to do it within 6 bits purely using the normal QUBO model, but this did inspire us to do a way of constraining our QUBO, since we did not have the first row and the final 2 rows, meaning we only need a 2x3 matrix to describe the system. Using logical statements, we can say which combinations lead to certain weights in use, requiring only 2 qubit gate models. The method we did this in is described below:

Our solution (constrained QUBO without last row):

Based on the provided information, we can summarize the approach to solving the Travelling Salesman Problem (TSP) using QAOA and adiabatic quantum computing as follows:

Formulate the TSP as a Quadratic Unconstrained Binary Optimization (QUBO) problem. For a 5-node TSP, we initially create a 5x5 matrix, where each row represents a step in the path traversal and each column represents a node. To ensure we visit each node only once and maintain a single-node visit per step, we can reduce the matrix to a 3x3 matrix and eventually use 6 qubits for 5 nodes.

Apply weights to each of the edges in the graph. Determine which edges were traversed by comparing every two adjacent rows in the matrix.

Calculate the total cost of the solution by summing up the weights of the traversed edges.

Add penalties for invalid moves:

a. For traversing a non-existent edge.

b. For visiting the same node twice - ensure each column has only one '1' (except for the last invisible row).

c. For being in two nodes at the same time - ensure each row has only one '1'.

Use the Quantum Approximate Optimization Algorithm (QAOA) or adiabatic quantum computing to find the solution with the lowest cost. QAOA is a quantum algorithm that approximates the ground state of a given cost Hamiltonian, which corresponds to the optimal solution in our case.

Decode the solution obtained from the quantum algorithm to determine the optimal path for the Travelling Salesman Problem. We will also see that the second lowest cost relates to the second best solution, etc..

3.Hardware Challenge:

In this hardware challenge, we have explored the system of 4 oscillators, linked in a circular way. We wanted to tackle the max cut problem using quantum annealing on Qibo.

We obtain the Ising coefficients via pair-wise Schrieffer-Wolff method.

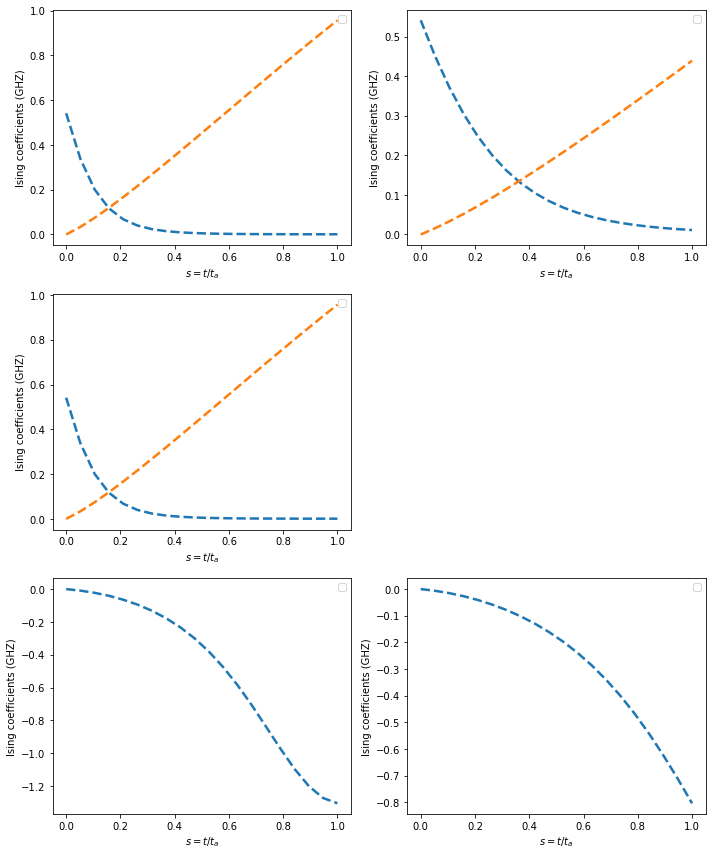


Fig 1: Ising coefficients calcualte via these two methods, where the solid lines are calculated using full SW, while dashed lines are calculated using the pair-wise SW method

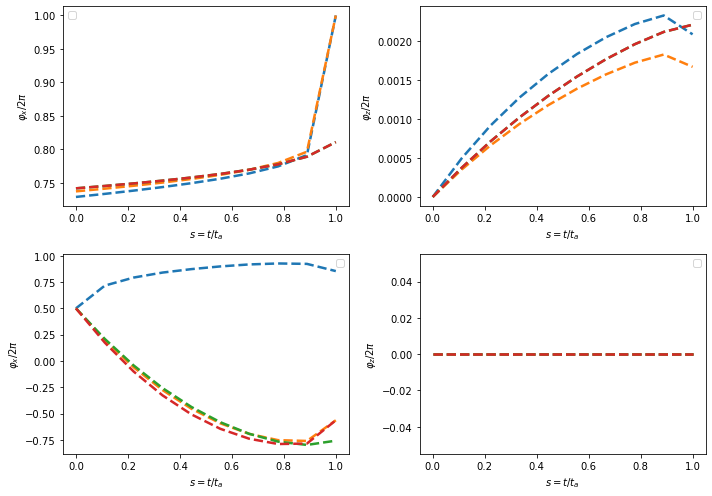


Fig 2: The extracted circuit biases using these methods, where the solid lines are calculated using the numerical method, while the dashed lines are calculated using the pair-wise method.

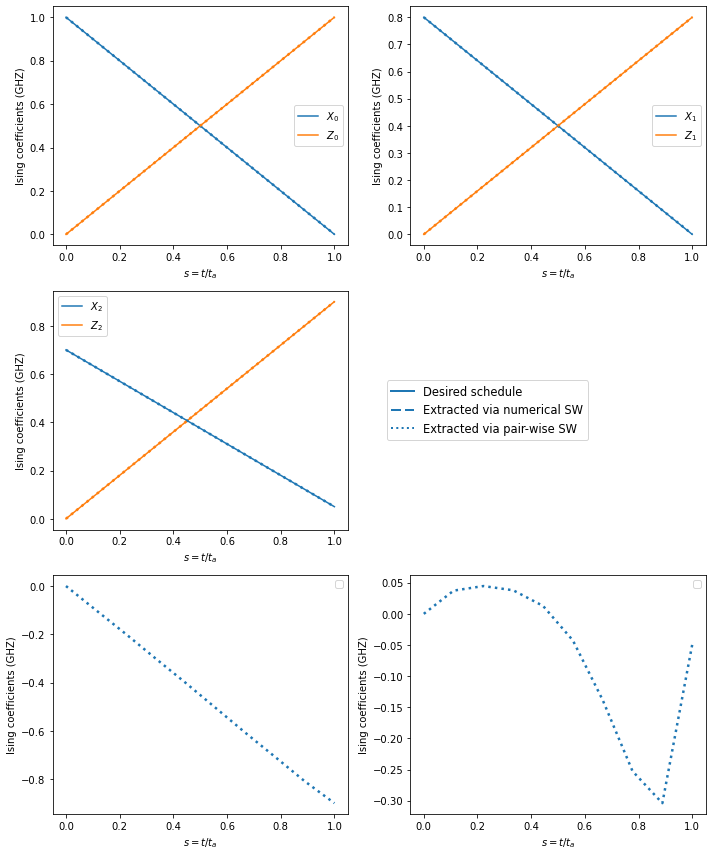


Fig 3: The schedules that our extracted circuit biases produce, and plot them against our desired schedules