The essence of problems where quantum computers fail to outperform classical computers

Joel Mathew Cherian (B190664CS)[1], Pavithra Rajan (B190632CS)[1], Nithin Puthalath Manoj (B190645CS)[1], and Adwaith Ram Kishore (B190808CS)[1]

[1]Department of Computer Science and Engineering, National Institute of Technology, Calicut

Abstract: The advent of quantum computation has revolutionised the field of computer science in numerous ways. By leveraging the rules of quantum mechanics to process information, quantum computers have proved to be exceptionally fast for a specific set of tasks. Unfortunately, for most of the problems, they can outclass classical computers only trivially. This project will focus on understanding the numerous complexity classes, the classes of problems solvable by both classical and quantum computers and delve deeper into the types of problems that quantum computers cannot significantly outperform classical computers. The latter will be dealt by taking a specific instance of such a problem to analyse the impact of solving it via a quantum approach. The real life implications of these problems will also be discussed.

1 Introduction

The ability of quantum resources to process information more effectively is universally recognized. While study predicts that for the majority of issues, quantum computers will only marginally outperform conventional ones, they would handle some specialized problems, such as factoring numbers, significantly quicker than we now know how to tackle them with today's computers [1]. Understanding which problems quantum computers can tackle more quickly than classical computers and how much quicker they can be is one of the objectives of quantum computing research. Two well-known quantum algorithms, Grover's and Shor's, outperform their conventional equivalents by polynomial and exponential factors, respectively. Hence, giving a quadratic speed-up.

Computer scientists categorize problems into overlapping classes based on the time complexity required to solve them. Four such classes are:

1. **P problems**: P is often a class of computational problems that are easy to solve in polynomial time and are tractable. Tractable means that the problems can be solved in theory as well as in practice.
2. **NP problems**: The solutions of the NP class are hard to find since they are being solved by a non-deterministic machine but the solutions are easy to verify. NP problems can be verified by a Turing machine in polynomial time.
3. **NP-Hard problems**: An NP-hard problem is at least as hard as the hardest problem in NP and it is the class of the problems such that every problem in NP reduces to NP-hard. All NP-hard problems are not in NP.
4. **NP-Complete problems**: A problem is NP-complete if it is both NP and NP-hard. NP-complete problems are the hard problems in NP. If one could solve an NP-complete problem in polynomial time, then one could also solve any NP problem in polynomial time.

Any computing problem that falls within the cat-

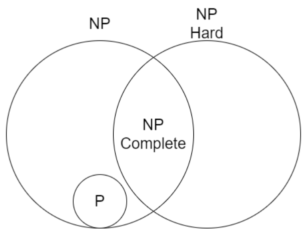


Figure 1: Classification of Problems

egory of NP-complete problems has yet to find an effective solution algorithm. This category includes a number of important computer science problems, such as the traveling salesman problem, satisfiability problems, and graph colouring problems.

In graph theory, graph colouring is a specific example of graph labeling. Vertex colouring is the process of colouring a graph's vertices so that no two neighbouring vertices have the same colour. 3 colouring problem is a subset of the said graph colouring problem where we can only use 3 colours.

For various NP-complete problems, a brute force approach is observed to be the fastest classical algorithm. For some, a refined brute-force techniques like backtracking can be adopted. Even though upper-bound time remains the same, the average time taken reduces giving us a time complexity of $O(3^n)$, where n is the number of vertices. Backtracking to solve the graph colouring problem is done as follows

- Assign colours to each vertex one by one
- Before each assignment, verify if any of the adjacent vertices have the same colour
- If there are no violations proceed else backtrack a step and reassign colours

It is not yet clear how best to encode in quantum computing data structures that are most commonly used in classical computing, such as general graph, a doubly linked list, and so on [2]. This greatly adds to difficulties related to directly programming a quantum computer to solve graph problems. Therefore we look at other ways to solve it in a quantum computer.

One such method that we will be looking into is to help design effective algorithms known as SAT-Reduction method. This model maps computationally hard problems on a general circuit-model quantum computer via reduction. By using a core quantum solver designed to solve a single problem, many other problems may be benefited from the quantum speed-up, by using a reduction wrapper around the core quantum solver. In other words, finding one solution potentially gives us immediate quantum accelerators for all.

2 Application

Graph Colouring problem has various real life applications. For instance, a colouring algorithm is embedded in some compilers. It is also used for allocation of radio broadcast frequencies, register allocation and scheduling of exam slots. Suppose we wish to schedule the exams for the below mentioned subjects in the least number of days, we can convert it to a graph colouring problem and find the chromatic number for the same. Consider the following subjects with their subject codes.

CS4021	Number Theory and Cryptography
CS4028	Quantum Computation
CS4032	Computer Architecture
CS4027	Cloud Computing
CS4038	Data Mining
CS4035	Computer Security
CS4041	Natural Language Processing
CS4042	Web Programming

Suppose there are no students who take the following courses together.

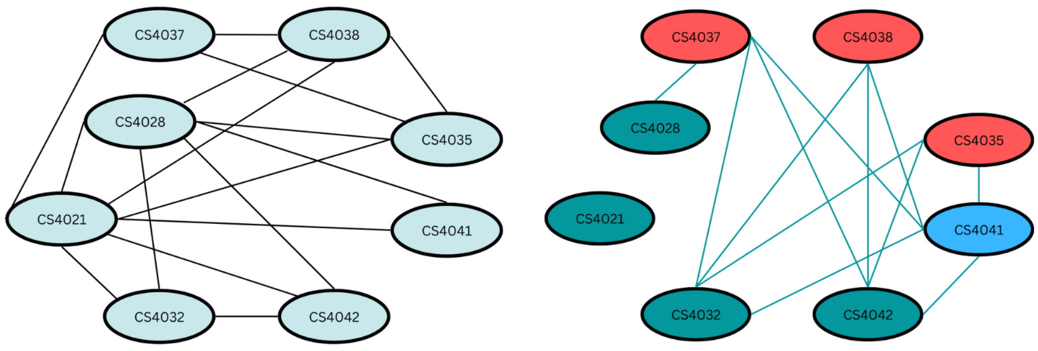


Figure 2: Exam slot scheduling via graph-colouring

Course Code	Not Taken With
CS4021	CS4028, CS4032, CS4037, CS4038, CS4035, CS4041, CS4042
CS4028	CS4032, CS4038, CS4035, CS4041, CS4042
CS4037	CS4038, CS4035
CS4032	CS4042
CS4038	CS4035

This problem can be converted to a graph colouring problem by representing the courses by vertices and connecting the courses with an edge if they have a student in common. In Figure 2, we can generate a graph and take the complement of it. We can then try to colour the graph with minimum number of colours.

- It is not 1-colourable as there are edges.
- It is not 2-colourable either as there are triangles.
- We can observe that it is 3-colourable and hence the minimum number of slots required is 3.

3 Problem statement

Though quantum computers can provide incredible speed-ups for specific kinds of problems, it cannot provide substantial performance speed-up for the class of NP-complete problems. Taking a specific instance of an NP-complete problem which is the 3-colouring graph problem, we will analyse the time-complexities of both classical and quantum approaches to determine if quantum computer can significantly outperform classical computers.

4 Literature Review

4.1 Chromatic number

At present, the fastest known classical approach to decide the k-colourability of an N-vertex graph is still exponential. In [3], the quantum algorithm can compute the chromatic number with a time complexity $O(1.9140^n)$. Though the classical approach takes $\Omega\ (2^n)$, the quantum approach doesn't give a significant boost in performance.

This approach uses **Byskov's algorithm** to compute the Maximal Independent Sets(MIS) of a fixed size. In graph theory, MIS problem revolves around finding the largest independent set in a graph. A set is independent if there is a set of vertices such that two vertices are not adjacent to each other. The backtracking method to find MIS is $O(2^n)$.

Additonally, we can change the k colouring to a $k^{'}$ colouring problem. If we take a subset of vertices S such that the graph with those S vertices it is $\lfloor \frac{k}{2} \rfloor$ colourable and the remaining graph excluding S is $\lceil \frac{k}{2} \rceil$ colourable, then it is k-colourable.

Using dynamic programming along with Grover's search on the MISs created by Byskov's algorithm, the chromatic number of the given graph can be determined. As the generation of MIS via Byskov's algorithm is a Branch and Reduce algorithm, we can apply Grover's search. A branching algorithm is an algorithm which recursively reduce a problem into problems with smaller parameters. If we have an MIS 'I' of size t and we choose T which is a subset of S excluding the nodes in MIS which satisfies $|T| < |S|/2$ and $|S - I - T| \leq \quad |S|/2$ then it implies that $|S|/2 - t \leq |T| \leq |S|/2$.

This is a sufficient condition to indicate that the chromatic number of the graph is correct. By analyzing the running time of this algorithm, the time complexity is $\mathbf{O(}1.9140^n\mathbf{)}$.

4.2 NK-method

In [4], a more generalised approach to colour a graph with N-vertices with K-colours has been introduced. Each node is given K qubits and a colouring matrix C which is N x K is defined. So for a valid C, there should be one 1 and K-1 zeroes as each node can have only one colour. The following steps computes the resultant matrix:

- Take the outer sum of the colouring matrix
- Concatenate the adjacency matrix of the graph K times
- Element wise product of the first two results

If we take the maximum from each row then it should be less than 2. That is, it should either be 0 or 1. The superpositions of all colouring matrices will be k^n. The time complexity in terms of number of qubits is $O(N^2)$, where N is the number of qubits. In terms of n and k it would be $\mathbf{O(}k^{2n}\mathbf{)}$, where n is the number of vertices and k is the number of colours. This approach is still in exponential time.

4.3 Reduction to a SAT problem

Quantum Computing is not experiencing the same growth as Quantum Computers and the widely believed statement that programming quantum computers is hard is backed up by solid evidence. To overcome this hurdle, we are introduced to an end-to-end framework that takes advantage of reductions [2]. It relies on the property that NP-Complete problems can be reduced to each other in polynomial time and therefore does not require quantum specific modeling and encoding techniques for various problems. This implies that if we can program a quantum computer to solve one particular problem, then we can apply that to speed up other problems.

The framework is outlined as follows:

- Implement a quantum algorithm that directly solves some particular problem S
- Transform the general input I to I' via polynomial time reduction
- Feed the input I' to S and get the output O'
- O' is converted back to the desired output O for the original input I

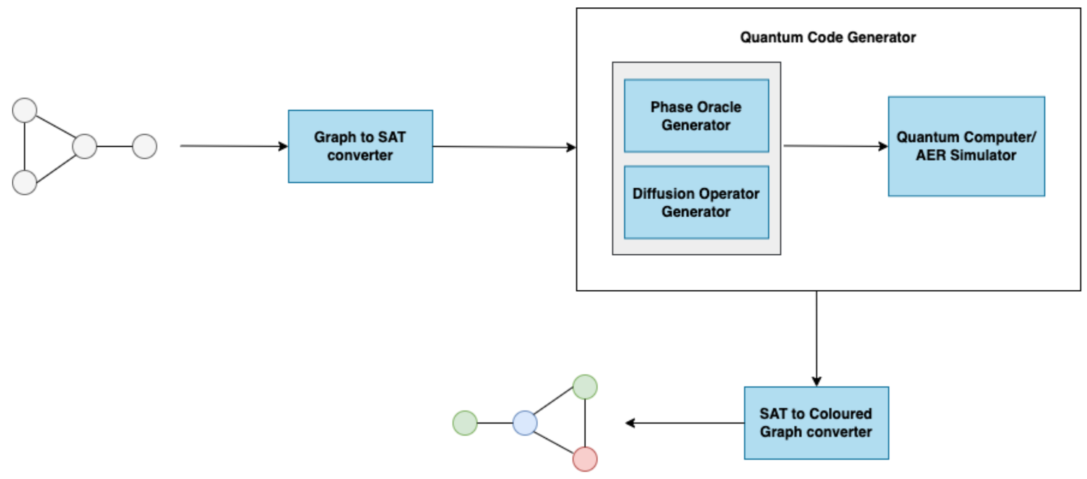
The unrestricted Boolean-SAT is chosen as the problem S, which can be solved with the help of Quantum Search (Grover's Algorithm).

The quantum search algorithm is setup in such a way to search in an unordered collection of N items. From a classical point of view, we need to make N/2 computations on average to find the item we are looking for. But from a Quantum stance, we can take advantage of superposition. Coupled with the Oracle function and Diffusion operator, we can leverage the Grover's Amplitude Amplification trick to search for the desired state inside the superposed state [5]. Compared to classical search, there is a significant speed up from O(N) to $O(\sqrt{N})$.

Briefly speaking, any input problem is transformed via reduction to a SAT instance, for which we use quantum search to find a solution that can then be transformed back for the original input according to the reduction.

5 Project Design

To understand the limitations of quantum computers in solving NP-complete problems in general and specifically 3-colouring problems, we have built a quantum-based 3-colouring problem solver using Qiskit. The solver can ascertain the 3-colourability of an input graph while generating possible solutions. The solver is built on the Reduction to SAT Problem approach [1]. This approach was adopted because it shows how NP-complete problems can be reduced to other NP-complete problems, 3-colouring to SAT in this case, and yet still



Quantum Code Generator
Graph to SAT converter
Phase Oracle Generator
Diffusion Operator Generator
Quantum Computer/ AER Simulator
SAT to Coloured Graph converter

Figure 3: Solver Design

suffers from exponential time complexity. There are three major modules: Graph to SAT converter, Quantum Circuit Generator, and SAT solution to Coloured graph converter. Figure 3 is a pictorial representation of the modules used in building the solver.

5.1 Graph to SAT

The Graph to SAT converter is the first module in the solver. It takes an adjacency matrix of a graph as input and outputs the SAT representation of the same problem. Consider a Graph $G = (V, E)$ where V is the set of vertices and E is the set of edges. The 3-colourability problem can be reduced into a SAT problem with variables v_i^R, v_i^G, v_i^B for every $v_i \in V$. For every vertex v_i the following boolean expression define that each vertex is colourable with only a single colour.

$$(v_i^R \vee v_i^G \vee v_i^B) \wedge \neg(v_i^R \wedge v_i^G) \wedge \neg(v_i^G \wedge v_i^B) \wedge \neg(v_i^R \wedge v_i^B)$$

For every edge $(v_i, w_j) \in E$, the following boolean expression ensures that no two adjacent vertices have the same colour.

$$\neg(v_i^R \wedge w_j^R) \wedge \neg(v_i^G \wedge w_j^G) \wedge \neg(v_i^B \wedge w_i^B)$$

For ease of generating the quantum circuit, the expression is restricted to use only the AND and NOT operators. Converting clauses to use only those operators can be done with relative ease. The expression is returned as a list of clauses, where each clause is represented as a list of variables. Whenever the NOT operator is to be applied, the $-$ sign is appended to the start of that variable. This SAT expression is passed as input to the Quantum Circuit Generator.

5.2 Quantum Circuit Generator

With the SAT expression as input, this module generates a quantum circuit implementing Grover's algorithm to return possible colourings for the graph.

The circuit consists of one qubit for each variable in the SAT problem. Additionally, a total of $2N - 1$ qubits are required, where one qubit is used to maintain the result of the SAT problem while the others help store intermediate results. The circuit starts by putting the variable qubits in uniform superposition by passing them through Hadamard gates. This output is then passed through a series of units that perform Grover's search on all possible states to identify the desired states. Finally, the variable qubits are measured to obtain possible solutions to the SAT problem.

All circuits implementing Grover's algorithm consist of a specific unit repeated in series. A unit consists of a phase oracle and a diffusion operator. These submodules are discussed below.

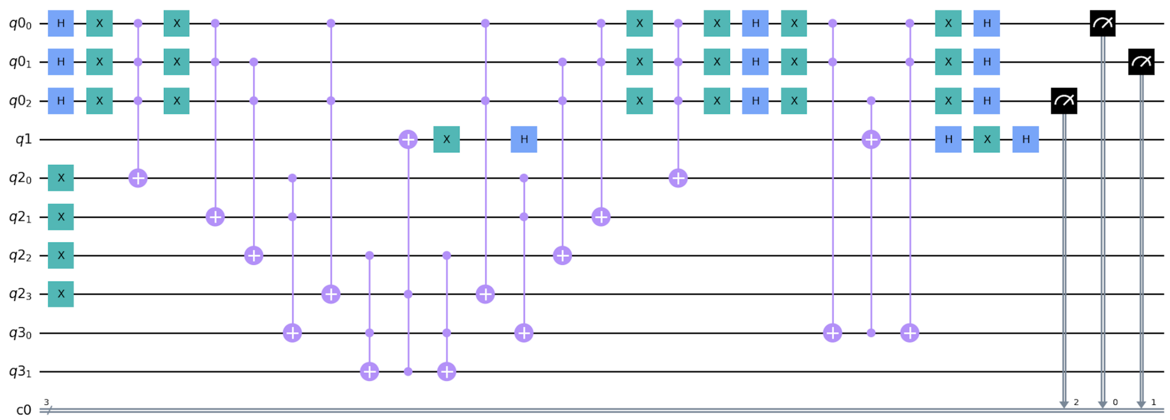


Figure 4: Quantum circuit generated by our solver for a trivial single node graph

5.2.1 Phase Oracle

The phase oracle is required to invert the phase of all the states that are being searched for. In this case, the oracle inverts the phase of all states that satisfy the SAT problem while leaving the others untouched. This can be achieved by implementing that SAT expression in the quantum circuit using X gates and Toffoli gates. When the AND of three variables are required in a clause a multi-control toffoli (MCT) gate can be used. However, it is unadvised to use an MCT for doing an AND of all the clauses. This is because such a gate will have a very large matrix that is quite hard to operate on since it comes with significant overhead. Instead, ancillae bits can be used to store intermediate results of each clause. When following this approach, the quantum circuit will have a V-shaped design. The output of the boolean expression is stored in an output bit. In creating the output signal, the variable qubits are entangled with the ancillae bits and we have to disentangle them in the reverse order. The output bit is then passed through an X gate. Note that the variables bits and the output bit still remain entangled after passing through the phase oracle. The output from the phase oracle for a N-variable SAT expression solver is an N+1 qubit system. Where the last qubit is 0 if the first N bits represent a solution to the SAT problem and 1 otherwise.

5.2.2 Diffusion Operator

The diffusion operator can be represented as M_n for n variable qubits [1].

$$M_n = H_n X_n Z_n X_n H_n$$

Here Z_n can be represented as an MCT of the n-variable qubits with the output bit sandwiched between two Hadamard gates. Again implementing a single MCT for n-variables adds a significant overhead, and instead, it is better to proceed with a composition of Toffoli gates. At the end, the output bit is passed through the X gate and a Hadamard gate to disentangle the output bit.

5.2.3 Repeating the circuit

The Grovers algorithm is such that the circuit has to be repeated a total of $\sqrt{(N/M)}$ times where N is the number of possible inputs and M is the number of solutions. Since N is exponential, the algorithm will take exponential time and hence does not significantly outperform classical computers. When building the circuit, the value of M is unknown. Hence we can proceed with a value that repeats the circuit a sufficient number of times to make the extraction of results possible. With every repetition of the circuit, the probabilities of the correct outputs increase. Figure 4 and Figure 6 are circuits generated by the quantum circuit generator.

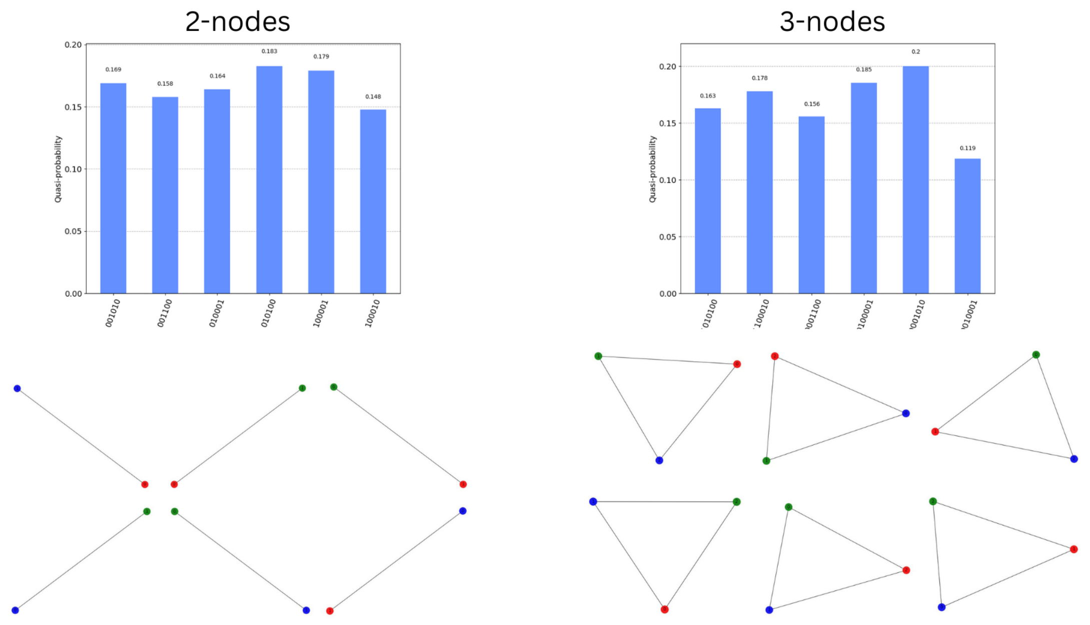


Figure 5: Results for two-node and three-node graphs

5.3 SAT to Coloured Graph

The output from the quantum circuit is a set of binary numbers; each number represents a solution to the SAT problem. Since each variable associated with a node indicates that node's colour, colours can be assigned to nodes, and possible outputs to the 3-colouring problem can be generated. Figure 5 displays the results produced by the solver for two-node and three-node graphs.

6 Work Done

Through the course of this project the following work has been done:

- Reviewed the most recent literature revolving the topic
- Documented the entire learning process as we realised one of the main challenges in this field is the lack of systematic documentation and comprehensive code structuring
- Implemented all of the components from scratch without the usage of pre-defined libraries to avoid black-box learning.
- Implemented a classical solution through backtracking for the 3-colouring graph problem
- Designed a scalable quantum approach to the problem which can be improved upon to generate more efficient solvers.
- Comprehensive analysis to prove the given problem statement
- Repository link: `https://github.com/JoelMathewC/Quantum-Computation-Project`

7 Conclusion

By taking an instance of an NP-Complete problem, we have seen that quantum computers do not significantly outperform classical computers. Any computational problem that can be solved by a classical computer can also be solved by a quantum computer. Conversely, any problem that can be solved by a quantum computer can also be solved by a classical computer with the principle that sufficient

time is given.
For all applications, a classic computer will not only continue to co-exist with quantum computers but also dominate in most of the arenas. It is not likely that we will be able to replace classical computers but rather have quantum computers complement them for certain tasks. Despite the fact that quantum computers may not be able to solve all the problems in a way that we expect it to do, it is not a reason for research to not be done in this domain.

References

[1] Aaronson, S. (2008). The limits of quantum. Scientific American, 298(3), 62-69.

[2] Hu, S., Liu, P., Chen, C. R., Pistoia, M., & Gambetta, J. (2019). Reduction-based problem mapping for quantum computing. Computer, 52(6), 47-57.

[3] Shimizu, K., & Mori, R. (2022). Exponential-time quantum algorithms for graph coloring problems. Algorithmica, 1-19.

[4] Clerc, M. (2020). A general quantum method to solve the graph K-colouring problem.

[5] D'Hondt, E. (2009). Quantum approaches to graph colouring. Theoretical computer science, 410(4-5), 302-309.

[6] Qiskit. (n.d.). Learn Quantum Computation using Qiskit. https://qiskit.org/textbook/

[7] NP-Complete problems `https://www.britannica.com/science/NP-complete-problem`

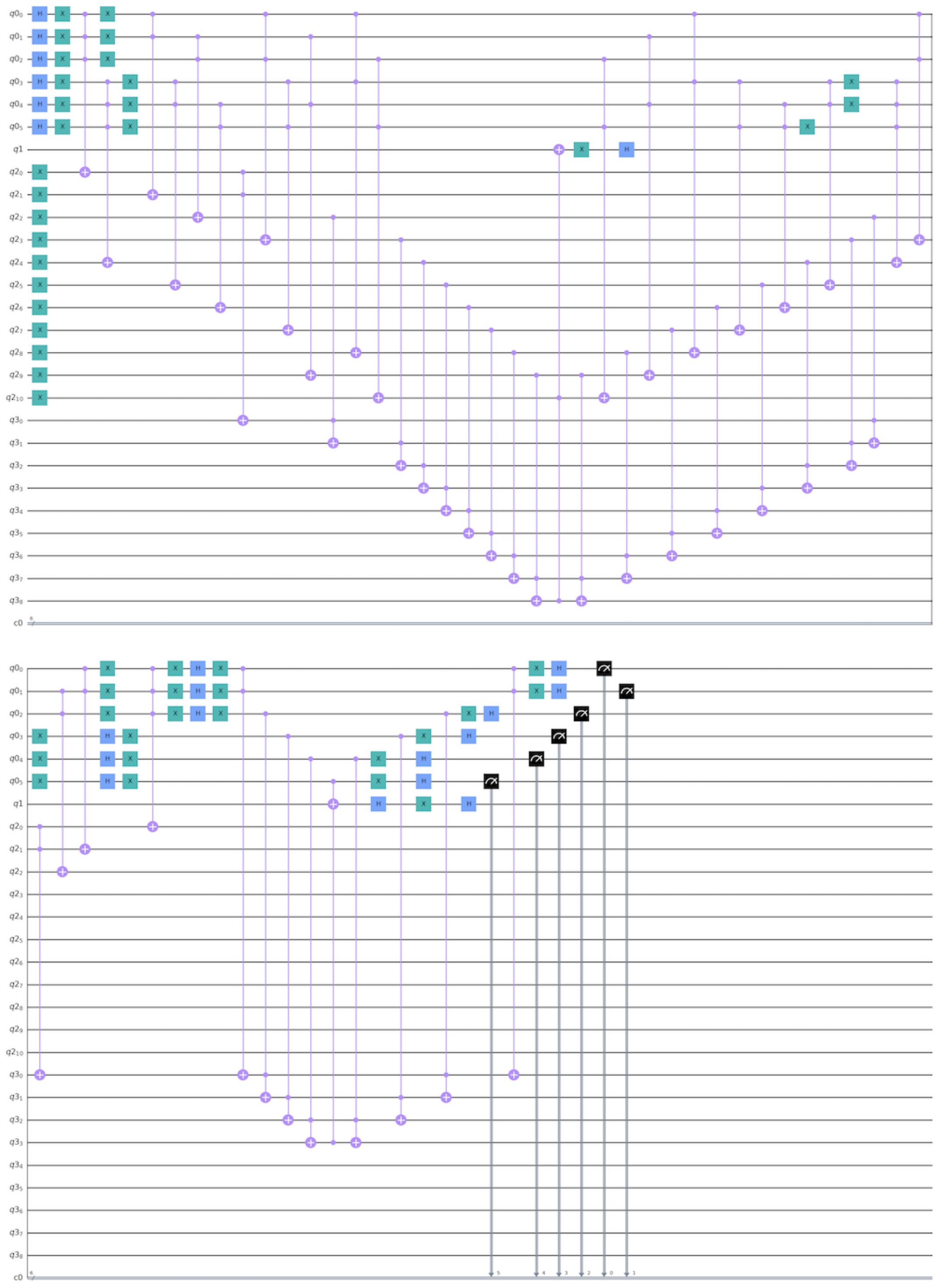


Figure 6: One unit of circuit generated for a two node graph