SOLVING 3-SAT PROBLEM USING A QUANTUM-SIMULATED ABSORBING CLASSICAL RANDOM WALK APPROACH

A Thesis

by

Alp Demirezen

Submitted to the
Graduate School of Sciences and Engineering
In Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in the
Department of Computer Science

Özyeğin University
May 2022

SOLVING 3-SAT PROBLEM USING A QUANTUM-SIMULATED ABSORBING CLASSICAL RANDOM WALK APPROACH

Approved by:

Professor Erhan Öztop, Advisor
Department of Computer Science
Özyeğin University

Assistant Professor Reyhan Aydoğan
Department of Computer Science
Özyeğin University

Professor Ahmet Celal Cem Say
Boğaziçi University
Department of Computer Engineering

Date Approved: 25.05.2022

To my family and friends who have always supported me throughout my journey.

ABSTRACT

Quantum computing offers novel approaches for solving computationally hard problems. In this thesis, we present a quantum algorithm based on the quantum simulation of Schöning's algorithm for solving the 3-SAT problem. We first introduce the concept of quantum-simulated classical absorbing random walk on a hypercube and illustrate the idea using Markov chains. Then we describe the quantum algorithm that is built on the mentioned concept for solving the 3-SAT problem. The algorithm starts by creating the equal superposition of all assignments to the variables representing the vertices of a hypercube. The next state is determined by querying the oracle that checks whether a clause is satisfied or not. Accordingly, one of the variables from an unsatisfied clause is flipped as in Schöning's algorithm. The resulting algorithm finds the solution with a probability that is equivalent to the expected success probability of Schöning's algorithm starting at all possible initial states. The algorithm uses a linear number of qubits in the number of variables provided that reset is possible and its performance is demonstrated through several 3-SAT instances. Its performance is compared to Grover's algorithm, and the proposed algorithm outperforms Grover's algorithm in most cases for the number of gates and depth.

ÖZETÇE

Kuantum hesaplama, hesaplama açısından zor problemleri çözmek için yeni yaklaşımlar sunar. Bu tezde, 3-SAT problemini çözmek için Schöning'in algoritmasının kuantum simülasyonuna dayanan bir kuantum algoritması sunuyoruz. İlk olarak, bir hiperküp üzerinde kuantum simülasyonlu klasik soğurucu rastgele yürüyüş kavramını tanıtıyoruz ve bu fikri Markov zincirlerini kullanarak gösteriyoruz. Daha sonra 3-SAT problemini çözmek için bahsedilen konsept üzerine inşa edilen kuantum algoritmasını açıklıyoruz. Algoritma, bir hiperküpün köşelerini temsil eden değişkenlere tüm atamaların eşit süperpozisyonunu oluşturarak başlar. Bir sonraki durum, bir tümcenin karşılanıp karşılanmadığını kontrol eden kahin sorgulanarak belirlenir. Buna göre, bir doyumsuz tümceden gelen değişkenlerden biri, Schöning'in algoritmasında olduğu gibi ters çevrilir. Ortaya çıkan algoritma, tüm olası başlangıç durumlarından başlayarak Schöning'in algoritmasının beklenen başarı olasılığına eşdeğer bir olasılıkla çözümü bulur. Algoritma, sıfırlamanın mümkün olması ve performansının birkaç 3-SAT örneği aracılığıyla gösterilmesi koşuluyla, değişken sayısında doğrusal sayıda kübit kullanır. Performansı Grover'ın algoritmasıyla karşılaştırılır ve önerilen algoritma, çoğu durumda kapı sayısı ve derinlik için Grover'ın algoritmasından daha iyi performans gösterir.

ACKNOWLEDGEMENTS

I would like to express my sincerest thanks to my advisors Prof. Erhan Öztop and Dr. Özlem Salehi Köken for granting me this opportunity and helping me throughout this journey whenever I need it. I am deeply grateful to the jury members, Prof. Cem Say and Assistant Prof. Reyhan Aydoğan who have agreed to participate in my thesis defense. I also would like to take this opportunity to thank my dear family, who always supported me with their endless and unconditional support. Finally, I would love to thank my closest friends for being there for me when I needed them the most.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Quantum computing offers novel approaches for solving problems that are computationally hard for classical computers. The most eminent examples can be listed as Shor's factoring algorithm [1], and Grover's search algorithm [2]. Grover's algorithm provides quadratic speed-up for any classical algorithm that uses brute force [3], yet there exist quantum algorithms with a better speed-up for particular problems. Hence, for which problems and to what extent quantum computing will bring advantage is still a field for investigation. The development of new techniques and tools and exploration of new algorithms are mutually advancing each other.

The problem of determining whether a logical statement consisting of boolean variables has a satisfying assignment or not is known as the *Boolean satisfiability problem* (SAT) problem. It is the first problem that is proven to be NP-Complete [4, 5]. k-SAT problem is a special case of SAT where each clause can have at most k literals. Due to its simplistic nature, a vast amount of research has been conducted for finding reductions from the other problems to 3-SAT and more efficient algorithms for the 3-SAT problem itself.

One of the most respected works regarding the k-SAT problem was due to Schöning [6]. The proposed algorithm suggests randomly creating an initial assignment and then creating new assignments with the guidance of the unsatisfied clauses. A new assignment is created by picking a literal from an unsatisfied clause and flipping the corresponding variable. After iterating the procedure for $3n$ times, the probability that a satisfying assignment is found is $(\frac{3}{4})^n$, where n is the number of variables. Hence, the algorithm has an expected runtime of $\mathcal{O}\left(\frac{4}{3}^n\right)$. An improvement to Schöning's algorithm is presented by Hofmeister et al. [7]

through a modification of the initialization step of the algorithm. Instead of starting with a random assignment, some assignments are eliminated depending on the clauses, improving the expected runtime to $\mathcal{O}(1.3302^n)$. Extending the initialization idea in [7], Baumer et al. proposed an algorithm with expected run-time $\mathcal{O}(1.3290^n)$ [8]. Some other randomized algorithms for 3-SAT following the same line of work include [9, 10, 11].

One of the first attempts in solving the 3-SAT problem using quantum computing was due to Ambainis. In [12], Ambainis showed that Grover's Search provides quadratic speed-up over Schöning's algorithm by replacing the classical repetition with the amplitude amplification procedure [3], thus obtaining a runtime of $\mathcal{O}(1.153\ldots^n)$. Barreto et al. replaced the local search in Schöning's algorithm with quantum spatial search and combined it with neighborhood search [13]. In addition, experimental results were provided by simulating the algorithm using parallel computing. Recent work by Dunjko et al. [14] offers a hybrid algorithm, which uses less quantum resources compared to [12] and provides polynomial speedup over Schöning's algorithm. The algorithm uses a divide and conquer approach, where the small quantum device comes into play for small instances.

In this work, we introduce a quantum algorithm for the 3-SAT problem that is obtained by simulating Schöning's algorithm quantumly. It is an application of the quantum-simulated absorbing random walk concept we introduce in this thesis. The idea is to use 'probabilistic coin flips' to create mixed states that simulate a classical random walk. Beginning at an initial superposition of multiple starting points, the absorbing states are reached with a probability that is equal to the average of the probabilities of a classical random walk starting at all possible initial states. We then enhance this idea by incorporating Schöning's algorithm into the picture. Basis states represent different bit assignments to the variables and at each iteration, one of the qubits that correspond to a literal from an unsatisfied clause is flipped. The resulting quantum algorithm provides the solution to the 3-SAT algorithm with a probability that is equivalent to the expected success

probability of Schöning's algorithm started at all possible initial states.

We run the algorithm on different 3-SAT instances and demonstrate its performance. The newly proposed quantum algorithm uses a linear number of qubits in the number of variables, provided that reset is possible in the quantum device. We also compare its performance with Grover's algorithm on the number of gates used and the depth of the circuit. The proposed algorithm outperforms Grover's algorithm in most cases for all metrics including the number of gates and depth.

The rest of the thesis is structured as follows. Chapter 2 presents the background information. Chapter 3 demonstrates the quantum-simulated classical absorbing random walk algorithm. Chapter 4 proposes the new algorithm for solving 3-SAT problem. In Chapter 5 we discuss our results. Finally, we conclude with Chapter 6 and give insights on future work.

CHAPTER II

BACKGROUND

This chapter will cover the topics necessary to follow up on this work including a brief introduction to quantum computing, Qiskit, the 3-SAT problem, and random walks.

2.1 Quantum Computing

We will cover quantum computing concepts in this section. For further information, we the refer reader to [15].

Classical computation utilizes bits to store information. A bit can exist in one of the states, either 0 state or 1 state. On the other hand, quantum computers store information in quantum bits, also known as qubits. A qubit has many different features from a bit however, one of the main reasons that they differ from each other is called superposition. The state of a qubit in superposition is described by a linear combination of the states 0 and 1. However, whenever it is observed, it is either in a 0 or a 1 state. Despite that, operations performed on the qubit in superposition prior to observation may affect the probability of 0 or 1 being observed when the qubit is measured.

2.1.1 Qubit

State 0 is represented as $|0\rangle$ and state 1 is represented as $|1\rangle$ in quantum computation. This notation is also known as the Dirac notation. A qubit can be expressed as

$$|\Psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$$

where α and β are complex numbers called the amplitudes. In this notation, the absolute squared value of the amplitude is equal to the probability of observing

the state, therefore, the equation $|\alpha^2| + |\beta^2| = 1$ must hold since the sum of the probabilities must be equal to 1.

A qubit can be also thought of as a two-dimensional unit column vector.

$$|\psi\rangle = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$$

Hence, ket notation $|.\rangle$ represents a column vector. One can use the bra notation $\langle .|$ to represent the complex conjugate transpose of the vector. For example,

$$\langle\psi| = \begin{bmatrix} \alpha^* & \beta^* \end{bmatrix}.$$

We use $\langle\psi|\phi\rangle$ to represent inner product and $|\phi\rangle\langle\psi|$ to represent outer product of vectors.

The inner product of any given quantum vector is equal to 1, therefore, we can state that the norm of any quantum vector is 1. The inner product is calculated as follows.

$$\langle\psi|\phi\rangle = \begin{bmatrix} \psi_1 & \psi_2 \end{bmatrix} \cdot \begin{bmatrix} \phi_1 \\ \phi_2 \end{bmatrix} = \psi_1\phi_1 + \psi_2\phi_2$$

For any given quantum vector, the inner product must be equal to 1. Below you can see the outer product calculation representation.

$$|\phi\rangle\langle\psi| = \begin{bmatrix} \phi_1 \\ \phi_2 \end{bmatrix} \cdot \begin{bmatrix} \psi_1 & \psi_2 \end{bmatrix} = \begin{bmatrix} \phi_1\psi_1 & \phi_1\psi_2 \\ \phi_2\psi_1 & \phi_2\psi_2 \end{bmatrix}$$

2.1.2 Mixed States

Quantum states described above are also known as pure states. Pure states are represented by ket vectors and satisfy all the mentioned properties. Suppose that we prepare the quantum state $|\psi_s\rangle$ with probability p_s. What we end up with is a mixed state, which is a statistical ensemble of the pure states $|\psi_s\rangle$. A mixed state can be expressed using a density matrix ρ which is obtained as follows:

$$\rho = \sum_s p_s \left|\psi_s\right\rangle \left\langle\psi_s\right| \tag{1}$$

2.1.3 Multiple Qubit Systems

A multiple qubit system represents the system that is obtained by combining multiple single qubit states. We express the resulting state mathematically using the tensor product. Let $|\psi\rangle = a_1 |0\rangle + b_1 |1\rangle$ and $|\phi\rangle = a_2 |0\rangle + b_2 |1\rangle$. The tensor product of the states $|\psi\rangle$ and $|\phi\rangle$ can be expressed as follows:

$$|\psi\rangle \otimes |\phi\rangle = (a_1 |0\rangle + b_1 |1\rangle) \otimes (a_2 |0\rangle + b_2 |1\rangle) \tag{2}$$

$$= a_1 a_2 |00\rangle + a_1 b_2 |01\rangle + b_1 a_2 |10\rangle + b_1 b_2 |11\rangle . \tag{3}$$

The symbol $\otimes$ is often omitted.

In a multi-qubit system that consists of n qubits, there are possible 2^n basis states and the corresponding state vector has 2^n entries.

2.1.4 Evolution

The evolution in quantum systems is linear. Quantum operators, which are square matrices, must preserve the norm of a given input vector and they need to be reversible due to the law of quantum mechanics. For an operator to be reversible, one should be able to tell the input of the operator just by looking at the output. For example, the classical AND operator is not a reversible operator, because we can not tell what the input was certainly when the given output is 0. The matrices corresponding to the quantum operators are known as unitary matrices. A matrix is unitary if;

$$UU^\dagger = U^\dagger U = I \tag{4}$$

where $U^\dagger$ refers to the conjugate transpose of U.

2.1.5 Quantum Circuit Model

The quantum circuit model is very similar to classical circuits. The main difference is that the quantum circuit model uses quantum logic gates to perform operations. Quantum gates are different from classical gates. However, we are in need of classical gates more than often because most of the problems we are trying to solve are constructed using classical logic. But it is possible to emulate the effects of classical gates by using quantum logic gates.

As mentioned above, quantum computers are not designed to store information in the same way as classical computers. Therefore, a different set of gates is needed to manipulate and use information in quantum computing. Each and every one of them can be represented by unitary matrices. We will be discussing some of the fundamental gates, and demonstrate their matrix representations.

2.1.5.1 Hadamard Gate

Hadamard gate can be used for creating an equal superposition of the basis states.

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

2.1.5.2 Pauli-X Gate

The Pauli-X gate or X gate is a single-qubit rotation through π radians around the x-axis. It is the NOT operator which maps state $|0\rangle$ to $|1\rangle$ and vice-versa.

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

2.1.5.3 Pauli-Y Gate

The Pauli-Y gate or Y gate is a single-qubit rotation through π radians around the y-axis.

$$Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$$

2.1.5.4 Pauli-Z Gate

The Pauli-Z gate or Z gate is a single-qubit rotation through π radians around the z-axis.

$$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

2.1.5.5 CNOT Gate

CNOT gate requires two qubits to be performed. One qubit would be the control qubit while the other one would be the target qubit. A Pauli-X gate is performed on the target qubit if the control qubit is in state $|1\rangle$.

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

2.1.5.6 Toffoli Gate

Toffoli gate is also known as Controlled Controlled NOT gate (CCNOT), it is similar to the CNOT gate, however, 3 qubits are needed to perform this operation. The first two qubits are control qubits and if both are in state $|1\rangle$, a Pauli-X gate is performed on the third qubit.

$$CCNOT = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

2.1.5.7 Multi-Controlled Gate Structures

Any of the given gates above can be implemented in such a way that they are controlled by multiple qubits. An example of a multi-controlled NOT gate can be seen in Figure 1.

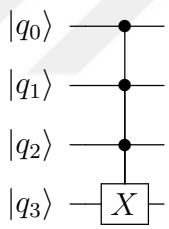


Figure 1: Exemplary multi-controlled NOT gate, where $|q_3\rangle$ is the target qubit.

2.2 Qiskit

Qiskit is an open-source framework for quantum computing provided by IBM [16]. Qiskit allows users to benefit from both IBM quantum computers and the simulator. Real quantum computers currently involve noise and they are limited in the number of qubits and connectivity. IBM quantum computers have a certain set of basis gates so that any quantum circuit is decomposed into that set of gates before it is run on the machine. This process is also known as transpilation.

2.3 3-SAT Problem

A logical formula is said to be in *Conjunctive Normal Form* (CNF) if it is expressed as a conjunction of clauses, where each *clause* is a disjunction of *literals*, each *literal* being either the variable itself or its negation. Deciding if there exists an assignment to the literals that make the logical formula true is known as the *satisfiability problem* (SAT).

3-SAT is a derivative of SAT problem where each clause is in CNF form and contains exactly 3 literals. 3-SAT is especially interesting since it is the derivation of SAT which is NP-hard with the lowest number of literals in a given clause. An instance of the 3-SAT problem consists of n variables $x_0, x_1, \ldots, x_{n-1}$ and k clauses. Given 4 variables x_0, x_1, x_2, x_3, below is an example of a 3-SAT instance with 3 clauses.

$$(x_0 \vee \neg x_1 \vee x_3) \wedge (x_0 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_0 \vee \neg x_1 \vee \neg x_2)$$

2.4 Random Walks

In this section, we will describe the idea of simulating a classical absorbing random walk on a hypercube using a quantum circuit. We will start with some background information on classical random walks and quantum walks.

2.4.1 Classical random walk

Consider an n-dimensional hypercube with 2^n vertices, whose vertices are labelled by $x_1x_2 \ldots x_n$ where $x_i \in \{0, 1\}$. Two vertices x and y are connected iff $\sum_{i=1}^{n} |x_i - y_i| = 1$, that is, if their Hamming distance is equal to 1. Now let's consider a random walk on this hypercube. Beginning at some initial vertex x_i, a particle moves to one of the neighboring vertices with probability $1/n$. We are particularly interested in *absorbing random walks*. A node x_i of the graph is an *absorbing state* if given that the particle is at x_i the probability that it stays at x_i is equal to 1. Suppose there is a single absorbing state, and the random walk starts at the vertex 0^n. It is proven that the expected time until the particle reaches one of

the absorbing states for a symmetric random walk on an n-dimensional hypercube which is known as the *absorbing time* is $\Theta(2^n)$ and that is independent of the location of the absorbing vertex [17].

2.4.2 Quantum walk

The randomness of Quantum walks is caused by a different reason from the classical random walks. The reason for the randomness of the classical random walk is the probability of reaching the next given state. Whereas the randomness in quantum walks is mainly caused by the superposition of the states. Quantum walk is not within the scope of this work, this section is mentioned for completeness.

Discrete-time quantum walks (DTQWs) introduced by Ahoronov et al. [18] are the quantum analogues of the classical random walks. Each vertex corresponds to one of the basis states of the quantum circuit. Single step of a DTQW is defined by $U = S \cdot (C \otimes I)$ where C is the coin operator and S is the shift operator. The coin operator is responsible for determining the direction of the walk. Selecting C as the Grover's diffusion operator [2] which is defined as $2\left|\psi\right\rangle\left\langle\psi\right| - I$ where $\left|\psi\right\rangle = \frac{1}{\sqrt{n}} \sum_{i=0}^{n-1} \left|i\right\rangle$, one obtains symmetric DTQW on the hypercube [19].

There are different ways to define absorbing time for quantum walks. In [19], absorbing time is defined as the probability that the absorbing vertex is eventually reached divided by the expected time to eventually reach the absorbing vertex. Through numerical simulation, it is shown that the absorbing time grows as $n^{1.5}$, exponentially faster than the classical walk when 1^n is the absorbing vertex.

The idea of symmetric DTQW is used to define a search algorithm on the n-dimensional hypercube where one of the vertices is marked, and the coin operator works as an oracle, behaving differently on marked and unmarked vertices [20]. The proposed algorithm makes $O(\sqrt{2^n})$ queries to the oracle.

CHAPTER III

QUANTUM-SIMULATED CLASSICAL ABSORBING RANDOM WALK

We will now consider the classical random walk algorithm implemented using a quantum circuit. We again assume that the vertices of the graph are represented by the basis states of the quantum circuit. For an n-dimensional hypercube, we construct a circuit with $n+1$ qubits where qubits $q_0, q_1, \ldots, q_{n-1}$ represent the vertices and q_n is the output qubit. We suppose that the oracle function $f_w(\cdot)$ takes as input a basis state and outputs 0 if the input state represents the marked vertex w, and 1 otherwise. $f_w(\cdot)$ can be also called with a state in superposition. The algorithm is presented below.

Algorithm 1 Quantum-simulated classical absorbing random walk

Input: Oracle function f_w
Generate an equal superposition of 2^n states
Repeat
  $q_n \leftarrow f_w(q_0, q_1, \ldots, q_{n-1})$
  $y \leftarrow$ random integer between 0 and $n-1$
  CNOT(q_n, q_y)

Let us investigate the steps of the algorithm in more detail. We start with the initial state $|0\rangle^{\otimes n}$, apply the Hadamard operator to each input qubit, and obtain the following state:

$$|\Psi_0\rangle = \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} |i\rangle |0\rangle . \tag{5}$$

Now we run the oracle function to check if the vertex is marked or not. This results in the following state:

$$|\Psi_1\rangle = \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} |i\rangle |f_w(i)\rangle , \tag{6}$$

where $f_w(i)$ is 1 if i is not the marked vertex and 0 otherwise.

At this stage, we apply a CNOT gate controlled by the output qubit, where the target qubit is picked randomly using a probabilistic coin flip. Let us illustrate it through an example. Suppose that $n = 2$ and 11 are the marked vertices.

$$|\Psi_1\rangle = \frac{1}{2}\big(|00\rangle|1\rangle + |01\rangle|1\rangle + |10\rangle|1\rangle + |11\rangle|0\rangle\big) \tag{7}$$

Assume that the first qubit is picked randomly. We apply CNOT and obtain the following state:

$$\begin{aligned} |\Psi_2\rangle &= \frac{1}{2}\big(|10\rangle|1\rangle + |11\rangle|1\rangle + |00\rangle|1\rangle + |11\rangle|0\rangle\big) \\ &= \frac{1}{2}|00\rangle \otimes |1\rangle + \frac{1}{2}|10\rangle \otimes |1\rangle + \frac{1}{\sqrt{2}}|11\rangle \otimes \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right) \end{aligned} \tag{8}$$

After applying CNOT, we reset the output register. Note that the probability of measuring state $|11\rangle$ has increased from $\frac{1}{4}$ to $\frac{1}{2}$.

The procedure can be summarized with the Markov chain given in Figure 2. Note that $|11\rangle$ is the absorbing state in this example. The states of the Markov chain are the different superpositions of the quantum state of the circuit. The transitions between the states take place as a result of the random coin flip. The initial state is the equal superposition state of the input register. Then with probability $\frac{1}{2}$, NOT gate is applied to either the first or the second qubit if the state is not $|11\rangle$. The amplitude of $|11\rangle$ increases or stays the same after any transition. Note that the described procedure above is different from the traditional DTQW that was explained previously. There are no additional coin states, and the evolution is taking place in mixed states. The key driver of the walk is the existence of the absorbing state.

In general, after the application of the first CNOT, the overall state $|\phi_1\rangle$ of the quantum circuit can be expressed by the following mixed state:

$$\rho_1 = \sum_{i=1}^{2} \frac{1}{2}|\phi_{1,i}\rangle\langle\phi_{1,i}|, \tag{9}$$

where $|\phi_{1,1}\rangle = \frac{1}{2}|10\rangle + \frac{1}{2}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$ and $|\phi_{1,2}\rangle = \frac{1}{2}|01\rangle + \frac{1}{2}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$. After the second step, the state is expressed as

$$\rho_2 = \sum_{i=1}^{4} \frac{1}{4}|\phi_{2,i}\rangle\langle\phi_{2,i}|, \tag{10}$$

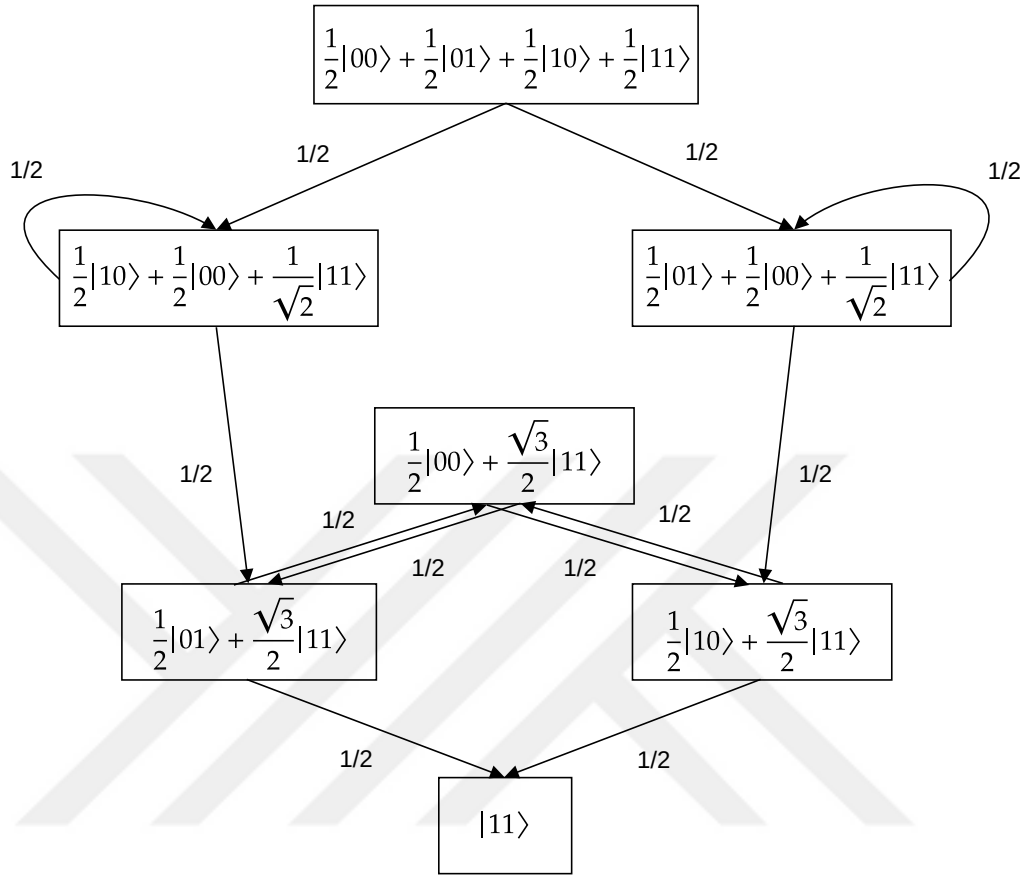


Figure 2: Markov chain representation of the quantum-simulated classical absorbing random walk procedure.

where $|\phi_{2,1}\rangle = \frac{1}{2}|10\rangle + \frac{1}{2}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$, $|\phi_{2,2}\rangle = \frac{1}{2}|01\rangle + \frac{\sqrt{3}}{2}|11\rangle$, $|\phi_{2,3}\rangle = \frac{1}{2}|01\rangle + \frac{1}{2}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$ and $|\phi_{2,4}\rangle = \frac{1}{2}|10\rangle + \frac{\sqrt{3}}{2}|11\rangle$.

If we generalize it to the case of n-dimensional hypercube, after step t, the state of the system is expressed by

$$\rho_t = \sum_{i=1}^{n^t} \frac{1}{n^t} |\phi_{t,i}\rangle \langle\phi_{t,i}|. \tag{11}$$

Each $|\phi_{t,i}\rangle$ represents the state that can be reached as a result of the random moves of the particle on the n-dimensional hypercube with $|1^{\otimes n}\rangle$ as the absorbing state, given that it starts in the equal superposition state.

On an n-dimensional hypercube, the classical random walk can start from any

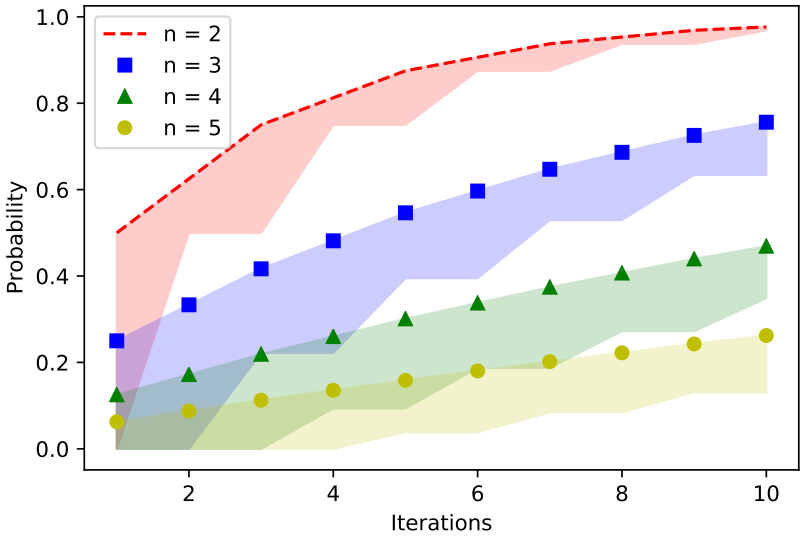


Figure 3: Probability that the absorbing state is reached after each iteration for the quantum-simulated classical absorbing random walk. The bottom of each shaded area corresponds to the classical random walk started at the initial state 0^n, which is the worst-case scenario. The marks correspond to the average case.

one of the 2^n states and the initial state plays an important role. For instance, starting in state 00, it is not possible to reach state 11 after 1 step but starting in state 10, the probability to reach state 11 is $\frac{1}{2}$ after one step. The quantum-simulated classical random walk simulates the classical random walk started in different initial states simultaneously. Assuming that the marked state is $|1\rangle^{\otimes n}$ and the computation starts in the equal superposition state, the probability to observe state $|1\rangle^{\otimes n}$ after t steps is exactly equivalent to the probability of observing 1^n averaged over all possible initial states after t steps. We will use the term expected probability to refer to the average probability of reaching a particular state using classical random walk, considering all possible initial states. In general, the diagonals of the density matrix corresponding to the mixed state ρ_t give the expected probability to reach each state after t steps.

The idea discussed above is plotted in Figure 3. The bottom of each shade denotes the probability to reach 1^n when the initial state is 0^n in a classical random walk, corresponding to the lowest probability. Note that the largest probability occurs when the initial state is 1^n, i.e. the trivial probability of 1 and this is not

depicted in the figure. The marks on the graph depict the probability to observe $|1\rangle^{\otimes n}$ using the quantum-simulated classical absorbing random walk, which also corresponds to the expected probability for the classical walk. Hence, we can conclude that using this approach where the main ingredient is superposition, we can simulate all branches of a probabilistic random walk simultaneously using a quantum circuit.

CHAPTER IV

A QUANTUM ALGORITHM FOR THE 3-SAT PROBLEM

In this chapter, we will describe a quantum algorithm for solving the 3-SAT problem, inspired by the random walk approach discussed in the previous section and Schöning's algorithm.

4.1 Schöning's algorithm

Schöning's algorithm [6] can be viewed as an improvement to the classical absorbing random walk on the hypercube. In the classical random walk, each step is taken randomly. This corresponds to flipping the value of one of the literals at random. In Schöning's algorithm, first, a clause that is not satisfied by the assignment is picked, and then a literal from that clause is flipped. The algorithm is presented in Algorithm 2.

Algorithm 2 Schöning's algorithm [6]

Input: A 3-SAT instance ϕ with n variables and k clauses
Guess an initial assignment $a \in \{0,1\}^n$
Repeat for $3n$ ***times***:
  if ϕ is satisfied **then**
    Stop
  $C \leftarrow$ a clause not satisfied by the assignment
  $x \leftarrow$ a random literal from C
  Flip value of x

Given a 3-SAT instance, let a^* be the satisfying assignment. Schöning's algorithm can be viewed as a Markov chain whose states are labeled by the particular state's Hamming distance to the solution. There is a designated start state from which there is a transition to state j with probability $\binom{n}{j}2^{-n}$, representing the initial random assignment. Suppose that state 0 is the absorbing state corresponding to the solution. Then from each state j, there is a transition to state $j-1$ with

probability at least $\frac{1}{3}$, and to state $j+1$ with probability at most $\frac{2}{3}$. The probability that the satisfying assignment is reached in $3n$ steps is greater than or equal to $\frac{4}{3}^n$.

4.2 Quantum-simulated classical absorbing random walk algorithm for 3-SAT

Combining the quantum-simulated absorbing random walk approach with Schöning's algorithm, we present the quantum algorithm in Algorithm 3. What we end up with is a quantum simulation of classical Schöning's algorithm, where instead of a single initial state, all possible initial states are simulated simultaneously. We have a small improvement over Schöning's algorithm when selecting the randomly flipped bit, as we keep a list of random selections to avoid the same clause to be selected again as long as there are clauses that are not selected yet.

We start with an equal superposition of assignments to the variables and follow the idea in Schöning's algorithm. f_r is the oracle function that returns 0 if the clause C_r is satisfied and 1 otherwise. Given a 3-SAT instance with n variables, we will use the n-qubit register to represent the variables and a single qubit output register to store the output of the oracle function.

Algorithm 3 Quantum-simulated Schöning's algorithm

Input : A 3-SAT instance with n variables and k clauses
Generate an equal superposition of 2^n states
$r_{list} \leftarrow [\,]$
Repeat:
  $r \leftarrow$ random integer between 1 and k not in r_{list}
  $r_{list}.append(r)$
  $q_n \leftarrow f_r(q_0, q_1, \ldots, q_{n-1})$
  $x \leftarrow$ random literal from C_r
  CNOT(q_n,q_x)
  if length(r_{list}) $= k$ **then**
    $r_{list} \leftarrow [\,]$
  reset q_n

Let us investigate the steps of the algorithm in more detail. We start with the initial state $|0\rangle^{\otimes n}$ and apply Hadamard operator to each input qubit and end up

with the following state:

$$|\Psi_0\rangle = \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} |i\rangle |0\rangle . \tag{12}$$

In the next step, a random clause C_r which is not previously selected is picked and it is checked whether the variable assignment represented by i satisfies C_r or not, and this information is stored in the output register.

$$|\Psi_1\rangle = \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} |i\rangle |f_r(i)\rangle \tag{13}$$

Here $f_r(i) = 1$ if the assignment i does not satisfy C_r and 0 otherwise. Next, we randomly pick one of the variables that appear in C_r and apply a NOT operator to the corresponding qubit controlled by the output register. The output register needs to be reset and then the same procedure is repeated by picking another clause randomly. Note that resetting is available in some of the quantum processors like those provided by IBM, but alternatively, one can use a new qubit for the next iteration. It is important to note that once all the clauses are picked at least once, then the picked clause list is reset and the same clauses start to get picked again.

Let us express the state of the system using mixed states approach as in Section 3. In this discussion, we will not consider the steps in which no variable is negated. The qubit on which the NOT gate will be applied depends on the chosen clause and a clause is chosen randomly at each step. Hence at the first step, it is possible to apply NOT gate to one of the qubits in set $\kappa_1 = \{q_1, q_2, \ldots, q_{\kappa_1}\}$. Note that the probabilities may not be equal as the same variable may be shared with different clauses, leading to a higher probability of negation for the corresponding qubit. The general state of the system after the first application of the NOT gate is given by

$$\rho_1 = \sum_{i=1}^{|\kappa_1|} p_{1,i} |\phi_{1,i}\rangle \langle\phi_{1,i}|, \tag{14}$$

where $p_{1,i}$ is the probability that qubit q_i is negated at step 1 and $|\phi_{1,i}\rangle$ is the state reached after q_i is negated at step 1. Based on the the probabilistic choices taken at each step, the state of the system after t steps will take the form

$$\rho_t = \sum_{i=1}^{|\kappa_1||\kappa_2|\cdots|\kappa_t|} p_{t,i} |\phi_{t,i}\rangle \langle\phi_{t,i}|. \tag{15}$$

Note that there are $|\kappa_1||\kappa_2|\cdots|\kappa_t|$ possible probabilistic choices until step t and $p_{t,i}$ denotes the probability of each. Here $|\phi_{t,i}\rangle$ is the state reached at step t, as the result of the probabilistic choice enumerated by i.

Now let us consider classical Schöning's algorithm. Let $P(a,t,s)$ denote the probability of observing state s at time t given that initial state is a. This probability would be alternating within a range between $[0,1]$. If we were to simulate Schöning's algorithm starting from all possible one of the 2^n states, we would obtain the following expression for the expected probability of reaching state s at time t:

$$\tilde{P}(t,s) = \frac{1}{2^n}\sum_a P(a,t,s). \tag{16}$$

This expression also coincides with the probability to reach any given state s using the quantum-simulated Schöning's algorithm, since we start in equal supeposition of all 2^n basis states. Hence, the diagonal entry of the density matrix ρ_t corresponding to the solution of the 3-SAT instance equals the average probability that the solution is obtained using Schöning's algorithm after t steps (where clauses are selected randomly at each step and selected clauses are appended to a list as in the algorithm we proposed) started with all possible initial assignments to the variables.

4.3 Implementation details

We utilised Qiskit [16] quantum programming framework provided by IBM to implement the proposed 3-SAT algorithm. The code is available at [21]. Now we will give the implementation details of the different components of the algorithm.

Let us consider how one can verify whether an assignment to the variables is satisfying or not for a clause by a quantum circuit, which we call the clause oracle. We will demonstrate the idea using the following 3-SAT instance:

$$(x_0 \vee \neg x_1 \vee x_3) \wedge (x_0 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_0 \vee \neg x_1 \vee \neg x_2) \tag{17}$$

The circuit given in Fig. 4 demonstrates the oracle function for the first clause.

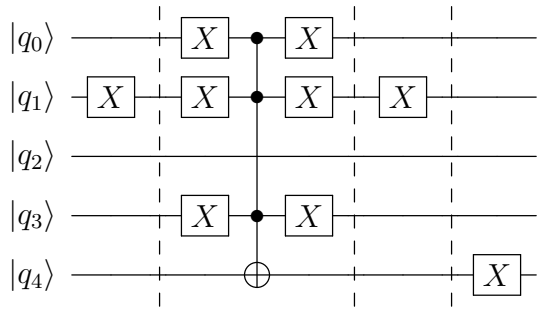


Figure 4: The oracle circuit sets $|q_4\rangle$ to $|1\rangle$ if the clause is not satisfied and to $|0\rangle$ otherwise.

The oracle function imitates the clause by applying NOT operators to the variables that are actually negated in the clause. Then, the function applies a NOT operator to the variables that are present in the clause it is inspecting. Later it performs a multi-controlled NOT operation to qubit q_4 with three control qubits. This series of operations can be thought of as the OR operation in classical computation. We uncompute by applying back the NOT operators to the variables that were initially negated. Finally, we apply a NOT operator to q_4 to output 1 if the result of the OR operation is 0, indicating the clause is not satisfied.

The random flip circuit given in Figure 5 aims to flip a random variable that is present in the clause if the current assignment is not satisfying. $|q_4\rangle$ is used as the control qubit for the CNOT operator with the target qubit being the randomly chosen variable from the clause. Finally, we apply a reset operation to have q_4 as a usable qubit in the upcoming iterations. Note that to avoid reset, one needs to use new qubits at each application of random bit flip.

Oracle circuits constructed using the idea described above are repeated for randomly picked clauses, aiming to find and correct the unsatisfied clauses. It is important to mention that the random bit flip circuit is only a demonstration as we do not know which variable will be flipped beforehand as it is picked randomly. The circuit given in Figure 6 shows the general structure. The oracle function described in 4 will be denoted O and the random bit flip circuit given in Figure 5 will be denoted by R and they are applied for m times where m denotes how

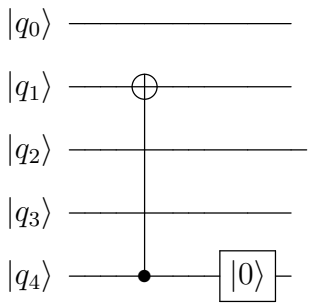


Figure 5: Random flip circuit flips the value of a qubit controlled by $|q_4\rangle$ and then resets $|q_4\rangle$.

many times this process will be repeated.

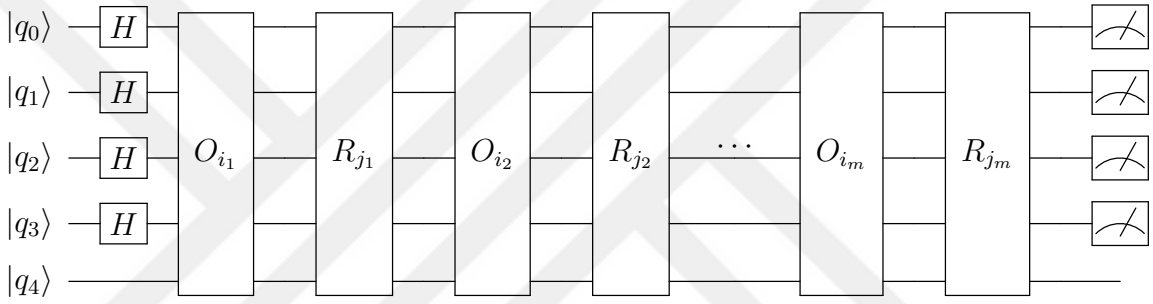


Figure 6: General structure of the circuit for solving the 3-SAT problem.

Assuming that the reset operation is available, the algorithm uses $n+1$ qubits where n is the number of variables. To implement multi-controlled NOT operation, the ancilla-free approach may be used from [22]. In this case, the depth of the circuit is $\mathcal{O}(nm)$, and the number of required gates is $\mathcal{O}(n^2m)$. If one uses $\mathcal{O}(n)$ ancilla [23], then the number of qubits will be still linear in the number of variables, and the circuit will have depth $\mathcal{O}(m \log n)$ and use $\mathcal{O}(nm)$ gates.

CHAPTER V

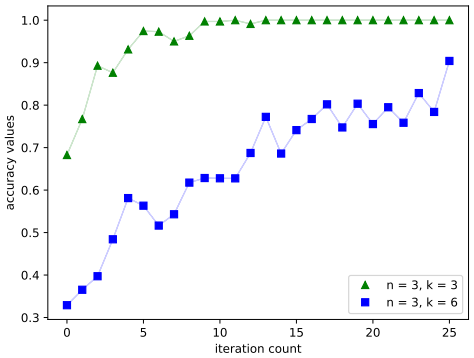
RESULTS

In this chapter, we present the results obtained by running the proposed algorithm on 3-SAT instances. We generated 50 different 3-SAT instances with a variety of n and k values, n ranging from 3 to 7 and k ranging from 3 to 13. 10 instances are generated for each n value. All of the results are obtained using the Qiskit simulator. In Table 1, the ratio of the number of satisfying assignments to the whole space 2^n is listed together with the number of iterations required to achieve 90% accuracy for some of the instances. The selected instances are picked so that we can demonstrate the importance of the satisfying ratio and how it is related to both n and k values. The information regarding the remaining instances can be found in the appendix.

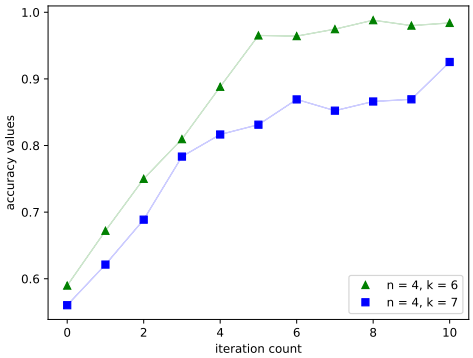
The accuracy of the proposed algorithm increases with the number of iterations as expected. Figures 7(a)-7(e) demonstrate the change in accuracy with respect to iterations for different instances. An instance that reaches the goal accuracy fast and another that reaches slowly is chosen for demonstration purposes for each n value. Each experiment is repeated 40 times and the results are averaged. The initial accuracy depends on the specific instance i.e., the ratio of satisfying assignments has a huge impact on how quickly the solution is found. Figure 7(f) demonstrates the number minimum and maximum iterations needed for different n values to reach 90% accuracy. Also, the figure illustrates that different iterations might be needed for the same n value.

Table 1: Satisfying assignment ratio and the number of iterations needed to reach 90% accuracy for each 3-SAT instance.

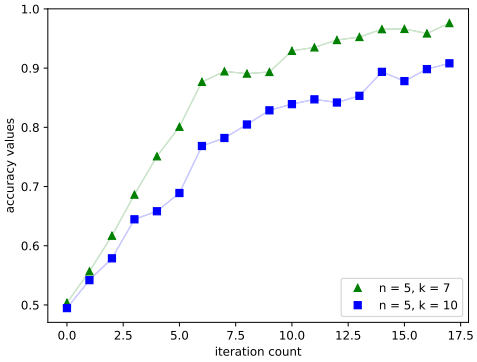
(n, k)	**(3, 3)**	**(3, 6)**	**(4, 6)**	**(4, 10)**	**(5, 7)**	**(5, 10)**	**(6, 8)**	**(6, 12)**	**(7, 9)**	**(7, 13)**
Ratio	0.62	0.25	0.50	0.38	0.44	0.34	0.31	0.36	0.33	0.28
Iterations	3	26	3	8	10	18	19	12	8	26



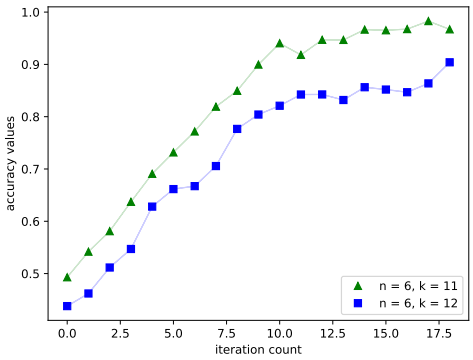
(a) $n = 3$.



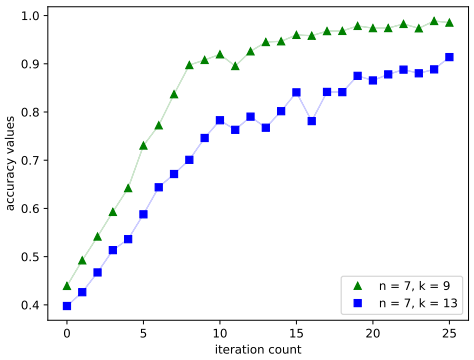
(b) $n = 4$.



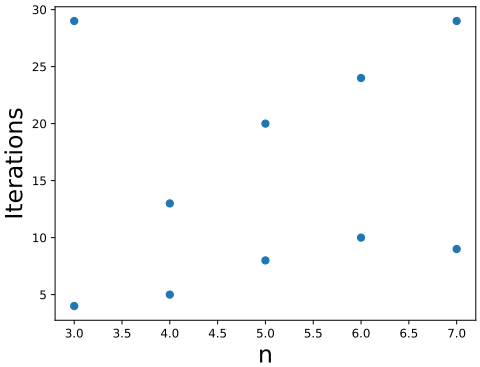
(c) $n = 5$.



(d) $n = 6$.



(e) $n = 7$.



(f) n vs. iterations.

Figure 7: In Figures (a)-(e), the change in accuracy with respect to the number of iterations is given for instances with different n values. The accuracy values are calculated by taking the average of 40 distinct calls to the algorithm. Among 10 instances for each n, two of the instances are picked for visualization. Figure (f) illustrates the minimum and maximum steps needed to reach %90 percent accuracy for different n values.

The iteration count increases depending on how hard it is to find a satisfying solution and increases as n increases as can be seen from Table 1. The ratio value demonstrates the possibility of finding a satisfying assignment with a random

guess. As n increases, the probability of finding a satisfying assignment by a random guess decreases as well, which is the main factor determining the required number of iterations. This is evident as different number of iterations are required for instances with the same n values.

The proposed algorithm uses an oracle to check whether each clause is satisfied or not. Another oracle-based approach for solving the 3-SAT algorithm is Grover's search algorithm. Proposed by Lov Grover [2], Grover's search algorithm solves the following problem: Given a function $f : \{0,1\}^n \mapsto \{0,1\}$, it finds out the inputs x for which $f(x) = 1$. The function f is the *oracle*. One can think of the domain of f as the search space, and those inputs x for which $f(x) = 1$ as the marked elements or the solutions. For instance, in the case of the 3-SAT problem, the search space consists of assignments to the binary variables, and the oracle checks if an assignment is satisfying or not and returns 1, iff it is a satisfying assignment. Given that there are $N = 2^n$ elements in the search space, Grover's algorithm requires $O(\sqrt{N})$ queries to the oracle and provides quadratic speedup compared to the best classical algorithm for the task.

Grover's algorithm consists of 2 phases implemented by oracle and diffusion operators. For the oracle, we allocate n qubits to cover each variable. For each clause i, the clause oracle O_i is applied where the target of the multi-controlled NOT operator is the $n+i$'th qubit. Finally, a multi-controlled NOT operator with the controls on the k qubits, and the target as the output qubit is applied. To change the sign of the marked elements, Z gate is applied on the output qubit. The multi-controlled NOT operator can be implemented using $k-1$ ancilla or no ancilla at all; however, note that the no-ancilla version requires more gates and depth. Depending on which implementation is used, we end up with a circuit using $n + k + 1$ and $n + 2k - 1$ qubits respectively, but let us note that more efficient oracle construction might be possible.

The diffusion phase is implemented through the standard application of Hadamard gates, NOT gates, and multi-controlled Z operator. Multi-controlled Z operator

is implemented by applying Hadamard to the target qubit, multi-controlled NOT, and applying Hadamard again. Based on the choice of Multi-controlled NOT operator, we again end up with two possible implementations.

We implemented Grover's algorithm using both the ancilla and no-ancilla versions of the multi-controlled NOT operator and compared them both with each other and with the proposed algorithm. One of the implementations needs an excessive amount of qubits to perform, whereas the other demands more gates and circuit depth. One could alternate between those two implementations to perform most efficiently in different scenarios. For the sake of demonstration, we have kept all of our instances within the qubit need of the highest demanding algorithm. However, it is important to mention that the proposed algorithm utilizes much less qubit resources compared to both implementations of Grover's algorithm. Our algorithm demands $n+1$ qubits to perform in Qiskit, whereas Grover's implementations demand $n+k+1$ and $n+2k-1$ respectively. We transpiled the circuits using the basis gate set of $\{Id, R_z, S_x, \text{CNOT}, \text{NOT}\}$, which is the set of basis gates used by IBM quantum computers [24]. Under these circumstances, the algorithms are run until reaching %90 accuracy and the depth and gate count comparison of the algorithms can be seen in Table 3. We have chosen those ten instances to be able to demonstrate the importance of the k value with a given n value and also, the importance of the n value itself. The gate count represents the count of the CNOT gates since they are the ones that heavily affect the performance of the circuit.

It is safe to assume that Grover implementations can slightly outperform the proposed algorithm when the initial success ratio is too low. (The ratio of satisfying assignments can be found in Table 1). However, in most cases, the proposed algorithm outperforms both implementations of Grover's algorithm for each metric. In addition, we can see that both implementations of Grover's algorithm need dramatically more resources for some particular instances. That is related to the

Table 2: Circuit depth and the number of CNOT gates are compared for the proposed algorithm and the two different implementations of Grover's algorithm for reaching %90 accuracy.

	Proposed Algorithm		**Grover (No ancilla)**		**Grover (Ancilla)**	
(n, k)	**Depth**	**#CNOT**	**Depth**	**#CNOT**	**Depth**	**#CNOT**
(3, 3)	116	35	372	192	259	96
(3, 6)	645	196	486	266	277	102
(4, 6)	116	35	2009	1096	1039	432
(4, 10)	231	70	9288	5478	1357	540
(5, 7)	277	84	3815	2100	1577	660
(5, 10)	346	105	3134	1848	434	186
(6, 8)	484	147	1196	696	345	156
(6, 12)	323	98	32927	19704	1309	684
(7, 9)	231	63	1969	1204	371	180
(7, 13)	645	196	21349	12836	1847	1008

needed iteration count in Grover's algorithm, which depends on the ratio of satisfying assignments. The main advantage of the proposed algorithm results from the oracle construction which checks whether a single clause is satisfied at a time, instead of checking the whole instance.

Finally, it is important to mention one major flaw of Grover's algorithm, regarding the 3-SAT problem. If the initial satisfying ratio is approximately %50, Grover's algorithm can not improve the accuracy. No matter how many iterations are run, the result will be approximately %50. Whereas the proposed algorithm does not have such constraint, and would reach the aimed accuracy value, regardless of the initial satisfying ratio, given the necessary amount of resources. The first instance with $n = 4$ and $k = 6$ values that can be seen from Appendix A is an example to this situation. One can read how many iterations are needed to reach %90 accuracy using the proposed algorithm. However, it is not possible to reach that accuracy with Grover's algorithm.

CHAPTER VI

CONCLUSION

To conclude, in this work we have proposed a quantum algorithm to solve the 3-SAT problem which is based on parallel simulation of Schöning's algorithm using a quantum circuit. We first presented a quantum circuit to simulate classical absorbing random walk on the hypercube. Combining the proposed approach with Schöning's algorithm, we presented the algorithm for solving 3-SAT. We simulated the algorithm on different 3-SAT instances and the results were promising as high-accuracy solutions were obtained using a small amount of resources. Also, it is observable that the proposed algorithm performs better than Grover's algorithm with respect to gate count and circuit depth for the same accuracy levels.

For the future work, we would like to compare the proposed algorithm with different quantum-based algorithms and observe how it can perform against them.

APPENDIX A

ALL TEST CASES AND GENERATED RESULTS FOR THE PROPOSED ALGORITHM TO REACH %90 ACCURACY

Table 3: Satisfying assignment ratio, the step count to reach %90 percent accuracy, depth of the circuit and the number of controlled NOT gates used are reported for each generated instance.

(n, k)	Ratio	Step	Depth	#CNOT
(3, 3)	0.625	5	116	35
(3, 3)	0.625	4	93	28
(3, 4)	0.5	11	254	77
(3, 4)	0.5	6	139	42
(3, 5)	0.38	16	369	112
(3, 5)	0.38	18	415	126
(3, 6)	0.25	28	645	196
(3, 6)	0.25	29	668	203
(4, 6)	0.5	5	116	35
(4, 6)	0.56	6	139	42
(4, 7)	0.38	13	300	91
(4, 7)	0.44	10	231	70
(4, 8)	0.44	8	185	56
(4, 8)	0.38	11	254	77
(4, 9)	0.38	9	208	63
(4, 9)	0.44	9	208	63
(4, 10)	0.38	10	231	70
(4, 10)	0.38	10	231	70
(5, 7)	0.44	12	277	84
(5, 7)	0.31	15	346	105
(5, 7)	0.44	8	185	56
(5, 8)	0.34	16	369	112
(5, 8)	0.47	8	185	56
(5, 8)	0.34	16	369	112

(n, k)	Ratio	Step	Depth	#CNOT
(5, 9)	0.34	16	369	112
(5, 9)	0.34	15	346	105
(5, 9)	0.31	15	346	105
(5, 10)	0.34	20	461	140
(6, 8)	0.31	21	484	147
(6, 8)	0.34	15	346	105
(6, 9)	0.37	24	553	168
(6, 9)	0.33	18	415	126
(6, 10)	0.30	19	438	133
(6, 10)	0.38	15	346	105
(6, 11)	0.28	22	507	154
(6, 11)	0.44	10	231	70
(6, 12)	0.36	14	323	98
(6, 12)	0.30	21	484	147
(7, 9)	0.33	10	231	70
(7, 9)	0.39	9	208	63
(7, 10)	0.38	10	231	70
(7, 10)	0.32	18	415	126
(7, 11)	0.28	21	484	147
(7, 11)	0.37	12	277	84
(7, 12)	0.23	29	668	203
(7, 12)	0.24	23	530	161
(7, 13)	0.28	28	645	196
(7, 13)	0.23	24	553	168

REFERENCES

[1] P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, pp. 124–134, IEEE, 1994.

[2] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, pp. 212–219, 1996.

[3] G. Brassard, P. Hoyer, M. Mosca, and A. Tapp, "Quantum amplitude amplification and estimation," *Contemporary Mathematics*, vol. 305, pp. 53–74, 2002.

[4] S. A. Cook, "The complexity of theorem-proving procedures," in *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, p. 151–158, Association for Computing Machinery, 1971.

[5] L. A. Levin, "Universal sequential search problems," *Problemy Peredachi Informatsii*, vol. 9, no. 3, pp. 115–116, 1973.

[6] U. Schöning, "A probabilistic algorithm for k-SAT based on limited local search and restart," *Algorithmica*, vol. 32, no. 4, pp. 615–623, 2002.

[7] T. Hofmeister, U. Schöning, R. Schuler, and O. Watanabe, "A probabilistic 3-SAT algorithm further improved," in *Proceedings of the Annual Symposium on Theoretical Aspects of Computer Science*, pp. 192–202, Springer, 2002.

[8] S. Baumer and R. Schuler, "Improving a probabilistic 3-SAT algorithm by dynamic search and independent clause pairs," in *Proceedings of the International Conference on Theory and Applications of Satisfiability Testing*, pp. 150–161, Springer, 2003.

[9] D. Rolf, "Improving randomized local search by initializing strings of 3-clauses," in *Electronic Colloquium on Computational Complexity*, 2003.

[10] K. Iwama and S. Tamaki, "Improved upper bounds for 3-SAT," in *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 328–328, Society for Industrial and Applied Mathematics, 2004.

[11] T. Hofmeister, U. Schoning, R. Schuler, and O. Watanabe, "Randomized algorithms for 3-SAT," *Theory of Computing Systems*, vol. 40, no. 3, pp. 249–262, 2007.

[12] A. Ambainis, "Quantum search algorithms," *ACM SIGACT News*, vol. 35, no. 2, pp. 22–35, 2004.

[13] M. Barreto, G. Abal, and S. Nesmachnow, "A parallel spatial quantum search algorithm applied to the 3-SAT problem," in *Proceedings of the XII Argentine Symposium on Artificial Intelligence*, pp. 1–12, 2011.

[14] V. Dunjko, Y. Ge, and J. I. Cirac, "Computational speedups using small quantum devices," *Physical Review Letters*, vol. 121, no. 25, p. 250501, 2018.

[15] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*. USA: Cambridge University Press, 10th ed., 2011.

[16] H. Abraham and the others, "Qiskit: An open-source framework for quantum computing (0.7.2)." `https://doi.org/10.5281/zenodo.2562111`, (Accessed May 2022).

[17] T. Yamasaki, H. Kobayashi, and H. Imai, "Analysis of absorbing times of quantum walks," *Physical Review A*, vol. 68, no. 1, p. 012302, 2003.

[18] D. Aharonov, A. Ambainis, J. Kempe, and U. Vazirani, "Quantum walks on graphs," in *Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing*, pp. 50–59, 2001.

[19] C. Moore and A. Russell, "Quantum walks on the hypercube," in *Proceedings of the International Workshop on Randomization and Approximation Techniques in Computer Science*, pp. 164–178, Springer, 2002.

[20] N. Shenvi, J. Kempe, and K. B. Whaley, "Quantum random-walk search algorithm," *Physical Review A*, vol. 67, no. 5, p. 052307, 2003.

[21] A. Demirezen, "alpdemirezen/quantum-3sat-solver: Initial release." `https://doi.org/10.5281/zenodo.6417084`, (Accessed May 2022).

[22] M. Saeedi and M. Pedram, "Linear-depth quantum circuits for n-qubit Toffoli gates with no ancilla," *Physical Review A*, vol. 87, no. 6, p. 062318, 2013.

[23] Y. He, M.-X. Luo, E. Zhang, H.-K. Wang, and X.-F. Wang, "Decompositions of n-qubit Toffoli gates with linear circuit complexity," *International Journal of Theoretical Physics*, vol. 56, no. 7, pp. 2350–2361, 2017.

[24] "IBM Quantum." `https://quantum-computing.ibm.com/`, (Accessed May 2022).