Computing, Performance and Communication Systems (2020) Vol. 4: 14-24
Clausius Scientific Press, Canada
DOI: 10.23977/cpcs.2020.41003
ISSN 2371-8889

Procedure of Solving 3-SAT Problem by Combining Quantum Search Algorithm and DPLL Algorithm

Runkai Zhang[1] , Jing Chen[2] and Huiling Zhao[2,*]

[1] *Miami college of Henan University, Henan University, Kaifeng, 475004, P. R. China*
[2] *School of Physics and Electronics, Henan University, Kaifeng, 475004, P. R. China*
**corresponding author*

Keywords: 3-SAT problems, Quantum search algorithm, DPLL algorithm.

Abstract: Although some classical algorithms have been applied to solve the satisfiability problem, more effective methods are still explored because existing algorithms are constrained by inadequate computing capability of traditional computers. The parallelism of quantum computation makes quantum algorithms with promising potential to improve the computing ability, but existing quantum algorithms still require too large number of qubits to solve a simple problem effectively. In this paper, an optimized data structure was structured to solve Boolean satisfiability problem by utilizing Grover's algorithm, and then the corresponding formula was proposed to balance variables in consideration of complexity. With reasonable simplification, quantum circuits were built to decrease the number of qubits required in Grover's algorithm. The result of verification experiment further demonstrated that the proposed approach is simple, reliable and of a certain practical value.

1. Introduction

The Boolean satisfiability problem (SAT problem) is the first problem that was proven to be Non-deterministic Polynomial complete (NP-complete) by Stephen A. Cook [1]. Before a problem can be solved by SAT methods, it should be transformed to the Conjunctive Normal Form (CNF): a conjunction of clauses, each clause being a disjunction of literals, and each literal being a Boolean variable or its negation. Among various SAT problems, 3-SAT is a famous one that has exactly 3 literals in each clause, which has been extensively investigated because it is a basic problem of Logic and Computer Science.

At present, the typical algorithms of solving SAT problems can be classified into two categories. One kind is called the Complete Algorithm which is mainly based on backtracking. Davis-Putnam-Logemann-Loveland algorithm (DPLL algorithm) is a crucial representative because truth assignment is adopted to simplify CNF [2]. The other method is called the Incomplete Algorithm and it is based on random search [3]. Most of these classical algorithms are limited by existing computing ability of classical computers. With the implementation of quantum computers and the research of quantum computing, parallelism and entanglement of quantum processing allow the

computation and communication capability to be improved. Subsequently, many new algorithms are proposed to solve 3-SAT problem.

There have been some results on how to use quantum gates to show if a CNF is satisfiable. Three quantum algorithms were reported by Alberto Leporati and Sara Felloni to test the satisfiability of CNF, in which one important method was the quantum Fredkin gate applied to solve 3-SAT problems [4]. Based on their research, Jin Wang et al. in Fudan University raised ETOF gate to test the satisfiability of given CNF [5]. These approaches are belonged to the Complete Algorithm because their research concentrate on the testing of satisfiability. However, in most cases, the aim of solving 3-SAT problems is to obtain the solutions of a Boolean function by exhaustive search on all its possible solutions.

The well-known algorithm called Grover's algorithm was firstly proposed by Lov K. Grover and used for searching the database [6]. Nowadays, there has been some research about applying Grover's algorithm to find solutions of Boolean satisfiability problem. The basic aspect of building conjunctions' circuits is how quantum gates used to express relations between variables, thus an approach of making quantum gates equivalent to classical gates is proposed [7]. In Figgatt's research, he presented how to solve the complete 3-SAT problem on a programmable quantum computer and gave examples of oracle built for correspondent solutions [8]. A circuit design for Grover's search algorithm by using 1-bit gates and 2-bit gates was presented by researchers in Texas A&M University [9]. In Diogo Fernandes's work, he summarized and carried out an approach to build the quantum circuits of Grover's algorithm [10]. Their results provide some instructive ideas for solving 3-SAT problems on quantum computer. A CNF always contain hundreds of variables and clauses for the most part. Therefore, a large number of qubits are required to obtain the solutions of CNF in general. On the other hand, utilizing qubits to calculate is not simple since all ensemble quantum computation should be carried out on sophisticated instruments rather than on a single computer. In such case, reducing the size of problems is beneficial because it improves the efficiency of solving SAT problems. By replacing quantum bits (qubits) with classical bits, Sheng-Tzong Cheng and Ming-Hung Tao presented a quantum cooperative algorithm to execute Grover's algorithm and classical algorithm simultaneously [11]. Although the mathematical verification proved it as an effective method, it is difficult to run two different types of algorithm on quantum computers and normal computers simultaneously. A. Montanaro introduced a universal method to execute quantum walk which is based on the technique of backtracking to obtain speedups of classical algorithms [12]. Based on Motanaro's research, Michael Jarret and Kianna Wan used effective resistance estimates to improve quantum backtracking and applied their algorithms on Constraint Satisfaction Problems [13]. However, the mathematical theories in their research are too complex to be employed in most practical programs.

In this paper, taking account of simplicity and uncomplicated implementation, we proposed a method that could divide a CNF into small parts and solve them separately by utilizing the structure of well-known DPLL algorithm. Due to its property of reducing the computing size of CNF, a formula was raised to compute the number of qubits used in quantum circuit. Finally, we verified the effectiveness of this method by using a frame called Qiskit.

2. Preliminaries

2.1. Set Representation

A function F of the SAT problem in CNF has the form like $F = c_1 \wedge c_2 \wedge c_3 \wedge \cdots \wedge c_n$, where each c_i is a clause formed by literals. In such case, a convenient way to represent a function is using sets. To be specific, a clause $c_i = l_1 \vee l_2 \vee l_3 \cdots \vee l_n$ can be described as the set $\{l_1, l_2, l_3, \cdots, l_n\}$, while the function $c_1 \wedge c_2 \wedge c_3 \cdots \wedge c_n$ can be described as $\{c_1, c_2, c_3, \cdots, c_n\}$. Thus, the function has been

converted from CNF into a set of sets. In our research, we constrained the problem to 3-SAT, where every clause has exactly three literals. For example, a set

$$S = \left\{\{A, B, \bar{C}\}, \{B, C, \bar{D}\}, \{C, E, F\}\right\} \quad (1)$$

is converted from

$$F = (A \vee B \vee \bar{C}) \wedge (B \vee C \vee \bar{D}) \wedge (C \vee E \vee F) \quad (2)$$

by using set-based notation.

2.2. Conditioning

Conditioning is an operation that replaces the literals in a clause with concrete value so that the initial formula can be converted into a new formula with fewer variables. The result of conditioning F on L is denoted by $F|L$ and it is described strictly below:

$$F|L = \{c - \{\bar{L}\} | c \in F, L \notin c\}. \quad (3)$$

To explain this formula specifically, every clause is a probability of three statements:

(1) Clauses include L. Clauses that mention L are automatically true since L is allocated to be true. In such case, these clauses have no impact on the satisfiability of the function so set of these clauses can be removed from $F|L$. Thus, the processed set does not include any clause that contains L.

(2) Clauses include $\bar{L}$. Since L is assigned to be true, then $\bar{L}$ is naturally false and the appearance of $\bar{L}$ does not influence the satisfiability of the clause that mentions it. Thus, the processed set $F|L$ has removed $\bar{L}$ from its clauses.

(3) Clauses include neither L nor $\bar{L}$. These clauses remain unchanged in $F|L$.

2.3. Termination Tree of DPLL Algorithm

To be brief, DPLL algorithm is a procedure that repeatedly operates conditioning and one way to judge whether the algorithm should be terminated is to measure the termination tree. In fact, the termination tree has two cases, and we can consider them separately. Firstly, the result of repetitive operation is a collection that includes an empty set {{}} due to three false literals in a clause. In that case, the CNF will never be true. Secondly, conditioning operation results in an empty set {} which means that all clauses are removed because they mention literals that are assigned to be true. If we use Δ to express the set after conditioning, then the termination tree can be expressed as:

$$\Delta = \begin{cases} \{\{\}\}, & Unsatisfiable\# \\ \{\}, & Satisfiable\#\# \end{cases} \quad (4)$$

2.4. Qubit

Compared to traditional bit that has a state which is either 0 or 1, qubit also has similar states. Two possible states for qubit are $|0\rangle$ and $|1\rangle$. Additionally, the most significant property of qubit is that the two states can exist simultaneously. Thus, a qubit can be expressed as a linear combination of two states:

$$|\varphi\rangle = \alpha|0\rangle + \beta|1\rangle, \quad (5)$$

where α and β are complex numbers. This is known as superposition in quantum mechanics, but a qubit will lose this property if it is measured. On the other hand, suppose multiple qubits are given,

their representation should be concerned. Let us assume a pair of entangled qubits is given and the formula is

$$|\omega\rangle = a|00\rangle + b|01\rangle + c|10\rangle + d|11\rangle, \tag{6}$$

where a, b, c and d are complex numbers. These four complex numbers are called amplitude and the probabilities to measure these four states are $|a|^2$, $|d|^2$, $|c|^2$ and $|d|^2$, respectively. The sum:

$$|a|^2 + |d|^2 + |c|^2 + |d|^2 = 1 \tag{7}$$

means that the result of a measurement must be one of these four states.

2.5. Grover's Algorithm

Grover's algorithm is one of the most important quantum algorithms, and its purpose is to find a specific element in a disordered array. If an array has $N = 2^n$ elements, Grover's algorithm only takes $O(\sqrt{N})$ operations to find a specific item due to the property of parallel computing. However, classical search algorithm must take $O(N)$ operations at the same case. Therefore, Grover's algorithm is competent for solving Boolean satisfiability problem.

Suppose a Boolean function F is given, Grover's search algorithm has two main operations to find the solutions.

1. Sign flip: Flip the signs of states which make $F = 1$.
2. Diffusion: Increase the amplitudes of states which evaluate $F = 1$ and decrease other amplitudes.

To be specific, we initially set n qubits to express n variables and apply Hadamard gate to create uniform superposition:

$$|s\rangle = H^{\otimes n}|0\rangle_n = \frac{1}{\sqrt[2]{2^n}} \sum_{x=0}^{2^n-1} |x\rangle_n \tag{8}$$

After that, we apply an oracle to flip the sign of states we want:

$$U_F|s\rangle = (-1)^F|s\rangle \tag{9}$$

where the value of F depends on the specific element. Then we apply the diffusion operator which can be represented as:

$$U_S = 2|s\rangle\langle s| - I_n U_S = 2|s\rangle\langle s| - I_n \tag{10}$$

where I_n represents identity matrix. Finally, the resulted state $|\varphi\rangle$ will be measured after t iterations and it is:

$$|\varphi\rangle = (U_S U_F)^t|s\rangle \tag{11}$$

3. Algorithm

We now introduce the as-proposed algorithm for solving 3-SAT problem and the entire algorithm process is presented in Figure 1. The main thought of this algorithm aims to reduce variables and clauses by conditioning some variables, to simplify the initial formula as much as possible. By following this strategy, a huge SAT problem can be divided into tiny parts which are easy jobs for a quantum computer to solve. To solve a Boolean satisfiability problem entirely, we have several steps below.

3.1. Selection of Variables

Since a Boolean formula can be extremely complex, we must choose appropriate number of variables to apply Grover's algorithm and some conventions should be made in advance.

Defeiniton 3.1.1. Any variable that appears in its original form like v_i is called its positive form, while the negation $\overline{v_i}$ of this variable is called its negative form.

Defeiniton 3.1.2. Any time a variable appears in a clause in its positive form or its negative form is defined as an occurrence.

Defeiniton 3.1.3. The weight w_i of any variable is defined as the total occurences of its positive form or negative form in an unprocessed Boolean function.

Thus, an ordered sequence may be obtained according to different weight w_i and variable v_i. Our strategy was to choose variables that have higher weight to operate conditioning because such variables result in simplification if it was eliminated. Assuming that the number of variables in a Boolean function was n and the number of variables calculated by Grover's algorithm was m, then the relationship between n and m can be expressed as:

$$m = \frac{2}{3}n + 2 - \frac{1}{3}\log_2 \pi^2 \tag{12}$$

Input: Boolean function F with n variables
Output: All solutions of F

```
Sort(Array)                                   ▷ Select appropriate variables
m = (2/3)n + 2 − (1/3) log_2 π^2
Conserve chosen variables
function SPAN(n_k, depth)                     ▷ Span the binary tree
    if depth = n − m then
        Conserve the path and return
    else if {} ∉ n_k then
        SPAN(n_k|v_{d+1}, depth + 1)
        SPAN(n_k|v̄_{d+1}, depth + 1)
    else
        Pruning
    end if
end function
while nodes in the bottom level do            ▷ Apply Grover's algorithm
    Quantum Search
end while
Combine to find general solutions             ▷ Find final solutions
```

Figure 1: Procedure of the proposed algorithm.

Here we give an explanation why m could be represented by the formula above. Since the depth of a binary tree spanned by DPLL algorithm increase rapidly with development of assigned variables, the number of nodes in the bottom level of binary tree may be enormous. Similarly, the

number of iterations also grows exponentially with the increase of qubits. Finding a minimum of the sum of these two terms is crucial. The function $f(m)$ was denoted as the sum of iterations and the depth, and it could be expressed as:

$$f(m) = \frac{\pi}{4} 2^{\frac{m}{2}} + 2^{n-m}. \tag{13}$$

Taking the derivative of $f(m)$ with respect to m, then the following equation was obtained:

$$f'(m) = \frac{\mathrm{d}f(m)}{\mathrm{d}m} = \frac{\pi}{8} 2^{\frac{m}{2}} \ln 2 - 2^{n-m} \ln 2. \tag{14}$$

$f(m)$ has a minimum when its derivative $f'(m)$ at a certain m is equal to 0. Solving the equation $f'(m) = 0$ and arranging the terms, we got the formula:

$$m = \frac{2}{3} n + 2 - \frac{1}{3} \log_2 \pi^2. \tag{15}$$

It should be noted that variables to operate conditioning must have higher weights than those which left to apply Grover's algorithm because these variables result in simplification to a greater extent.

3.2. Spanning of Binary Tree

By applying conditioning repeatedly, the result of repetitive operation was a binary tree which had some special properties below.

Property 3.2.1. The depth of first level is zero.

Property 3.2.2. Every node in the binary tree represents a set converted from CNF.

To illustrate these two properties, n_1 (Figure 2) is in the first level and the depth of this level is 0 since the node represents the original set that converted from given Boolean function.

Property 3.2.3. Every edge in the binary tree is an assignment of a variable.

Property 3.2.4. The assignment of a concrete variable leads to set of conjunctive normal forms in the same level.

Property 3.2.5. The left child of a node is the result of assigning a variable to be true, while the right child is the result of assigning a variable to be false.

However, we concentrated on pruning while we were building a DPLL tree. In our algorithm, we do not need to find the termination tree because quantum search algorithm helps to obtain solutions, but a collection that includes an empty set like $\{\{\}\}$ may occur during the process. If a similar case appears, the branch should be immediately cut off because the Boolean function will never be true along that branch, and it is meaningless to apply Grover's algorithm for building the simplified formula. If we use n_k to express the node after conditioning where $k \in (1,2,\dots,2^{n-m+1}-1)$, the two cases can be represented by:

$$n_k = \begin{cases} \{\{\}\}, & pruning \\ \{\}, & otherwise \end{cases} \tag{16}$$

3.3. Applying Grover's Algorithm

Note that if we wish to obtain all solutions of a given CNF, we need to apply Grover's algorithm to every node in the bottom level. While constructing the binary tree, some nodes that include empty set should be pruned. Thus, the times of applying Grover's algorithm is less than or equal to 2^{n-m}.

However, we found that the complexcity of quantum circuit was dominated by the specific set of CNFs. In other words, gates and qubits which were used to represent unknowns and relationships between variables were depended on the simplified degree of specific set. Since the assignments of simplification varied dramatically, simplified sets were various also and some of them even failed to belong to 3-SAT problems. We could also find that a CNF is not satisfiable if Grover's algorithm applied on each node do not show the probabilities as expected. In addition, the times of Grover's algorithm executed to obtain all the solutions and that of judging whether the CNF is unsatisfiable are equivalent, which is 2^{n_c} (without consideration of pruning) if n_c denoting the number of variables used in the spanning of binary tree. However, to test the satisfiability of a given CNF, only one time of Grover's algorithm might be enough in the best case.

3.4. Combination of Partial Solutions

If a branch is finished (depth is equal to $n-m$ and no {} is included in the set), the path of assignments must be conserved. Several solutions are found by Grover's algorithm, and it should be noticed that these solutions are partial. The entire solution is the combination of path, result of quantum search algorithm and arbitrary solutions. Since several sets are removed out due to one assigned literal included, left variables in the set will have no limitation so they are called arbitrary solutions. Thus, the combination of different solutions.

$$Soultions \overset{Sum}{\Longleftarrow} \begin{cases} Path \\ Partial\ Soultions \\ \text{Arbitrary solutions} \end{cases} \tag{17}$$

forms the entire solution of given Boolean function.

4. Experiment

In this section, we used an example to verify our algorithm. Given a CNF which had 9 variables and 10 clauses:

$$\begin{aligned} F = {} & \{v_1 \vee v_2 \vee v_4\} \wedge \{v_1 \vee v_2 \vee v_3\} \wedge \{v_1 \vee \overline{v_2} \vee v_3\} \wedge \{\overline{v_1} \vee v_3 \vee \overline{v_4}\} \\ & \wedge \{v_2 \vee v_3 \vee v_5\} \wedge \{\overline{v_5} \vee v_6 \vee v_7\} \wedge \{v_5 \vee \overline{v_6} \vee v_7\} \\ & \wedge \{\overline{v_5} \vee v_6 \vee \overline{v_7}\} \wedge \{v_2 \vee v_8 \vee v_9\} \wedge \{v_1 \vee v_3 \vee v_9\}\# \end{aligned} \tag{18}$$

It could be expressed as the set of sets below:

$$\begin{aligned} S = {} & \{\{v_1, v_2, v_4\}, \{v_1, v_2, v_3\}, \{v_1, \overline{v_2}, v_3\}, \{\overline{v_1}, v_3, \overline{v_4}\}, \\ & \{v_2, v_3, v_5\}, \{\overline{v_5}, v_6, v_7\}, \{v_5, \bar{v}_6, v_7\}, \\ & \{\overline{v_5}, v_6, \overline{v_7}\}, \{v_2, v_8, v_9\}, \{v_1, v_3, v_9\}\}\# \end{aligned} \tag{19}$$

According to equation (12), $n-m \approx 2.1$ variables should be used to span DPLL tree, here taking $\lceil n-m \rceil$ to get an integer aimed to use less qubits. Therefore, 3 variables should be used to construct the DPLL tree and their weight are shown in Table 1. In Procedure 1, it shows that the weight of every variable in the given Boolean function should be stored in the $array$, and then the $array$ needs to be sorted in order to make the selection of $n-m$ variables span a DPLL tree.

Table 1: Variables and their correspondent weights.

Variables	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9
Weights	5	5	5	2	4	3	3	1	2

Thus, v_1, v_2 and v_3 were chosen to span a DPLL tree which is depicted in Figure 2. Therefore, every node at level 3 was a formula that was simplified for Grover's algorithm except node n_{13} and n_{15}. From Figure 2, the node n_{13}:

$$\{\{\}, \{\overline{v_5}, v_6, v_7\}, \{v_5, \overline{v_6}, v_7\}, \{\overline{v_5}, v_6, \overline{v_7}\}, \{v_9\}\} \tag{20}$$

and n_{15}:

$$\{\{v_4\}, \{\}, \{v_5\}, \{\overline{v_5}, v_6, v_7\}, \{v_5, \overline{v_6}, v_7\}, \{\overline{v_5}, v_6, \overline{v_7}\}, \{v_8, v_9\}, \{v_9\}\} \tag{21}$$

both have empty set included. Hence, pruning was operated, and the simplified formula was meaningless to apply Grover's algorithm. We selected one node in the bottom level and converted the simplified set of CNFs into quantum circuit to test the satisfiability.

To express the process clearly, we chose the node n_8 in Figure 2 and a simplified set of CNFs was obtained, which is:

$$\{\{\overline{v_5}, v_6, v_7\}, \{v_5, \overline{v_6}, v_7\}, \{\overline{v_5}, v_6, \overline{v_7}\}\} \tag{22}$$

In this case, we constructed the circuit by using Qiskit and executed it on a simulator. According to De Morgan's law, each clause can be converted into a combination of disjunction and negations, moreover a clause can be represented as:

$$(v_1 \vee v_2 \vee v_3) \Leftrightarrow \left(\overline{\overline{v}_1 \wedge \overline{v}_2 \wedge \overline{v}_3}\right) \tag{23}$$

Thus, each clause in quantum circuit was represented by using Toffoli gate and Pauli-X gate. Pauli-X gate is simple since it is like classical NOT gate, while Toffoli gate is different. Toffoli gate consists of two control bits and one target, it flips the target only when two control bits are true. We chose the first clause as an example and the partial circuit in Qiskit is illustrated in Figure 3.

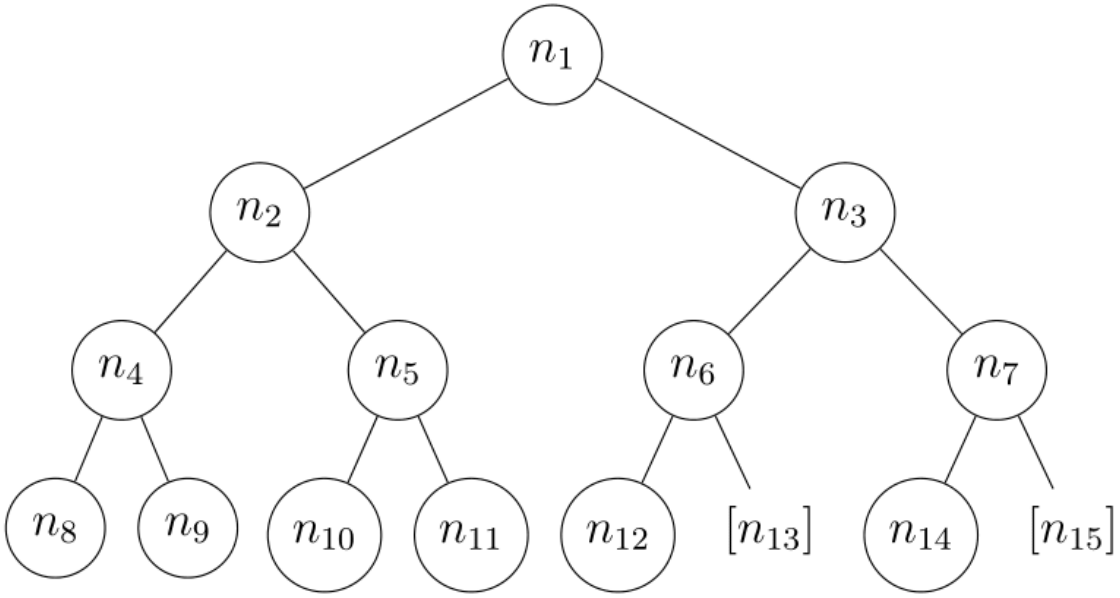

Figure 2: DPLL tree spanned by three nodes.

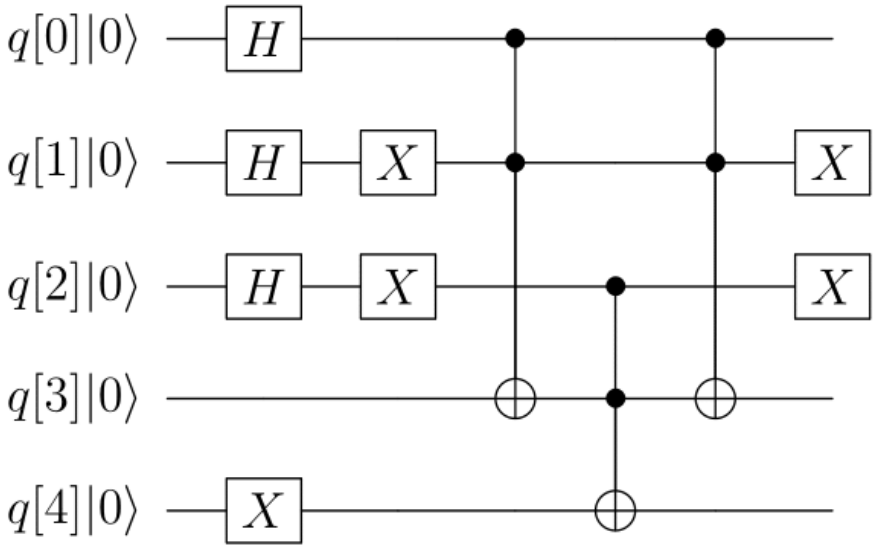

Figure 3: Three Hadamard gates are placed to create superposition and q [4] is flipped if q [0], q [1] and q [2] are true.

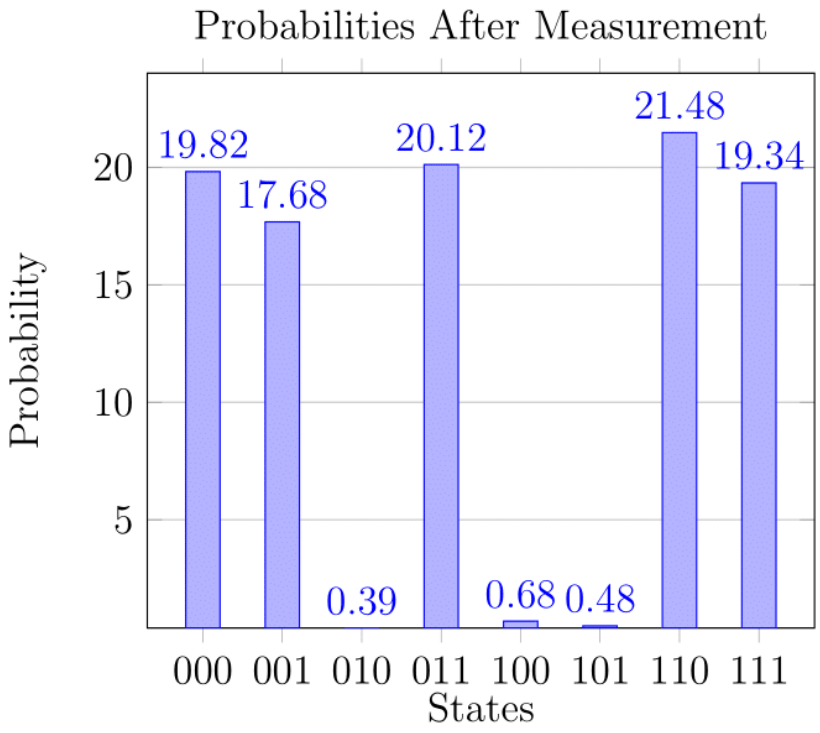

Figure 4: After measurement, only 010, 100 and 101 are not solutions for given simplified set.

We measured the three qubits which represented v_5, v_6 and v_7, and the result demonstrated that some assignments had higher probabilities while some had lower probabilities. Hence, solutions arose according to that higher amplitude means higher probabilities in measurement. For this example, the result as depicted in Figure 4 so that we can easily distinguish if an assignment is a solution. Since several solutions were found by Grover's algorithm, it should be noticed that these solutions were partial. Thus, the combination of different solutions is:

$$Soultions \overset{Sum}{\Longleftarrow} \begin{cases} Path: v_1 = 1, v_2 = 1, v_3 = 1 \\ PartialSoultions: v_5, v_6, v_7 = \{\{0,0,0\}, \dots\} \\ Arbitrarysolutions: v_4 = 0 \text{ or } 1 \end{cases} \tag{24}$$

If we apply Grover's algorithm again to every node in the bottom level except n_{13} and n_{15}, the entire solutions of the given Boolean function will be obtained.

5. Conclusions

In this article, we proposed a combined method to solve Boolean satisfiability problem through considering simplicity and implementation. By applying the structure of DPLL algorithm, we also raised a formula to compute the number of qubits used in quantum search. With rational simplification, a CNF was divided into small parts and quantum circuits were built. Finally, we verified the algorithm by using Qiskit. This work provides some specific hints for the development of quantum computing, and the proposal quantum algorithms to solve the Boolean satisfiability problem by simpler path and with less computing cost.

Acknowledgements

This research is funded by the Innovation and Entrepreneurship Training Program for College Students of Henan Province. The grant number is S202010475117.

References

[1] S. Cook (1971) The complexity of theorem-proving procedures. 151–158.
[2] M. Davis, G. Logemann, and D. Loveland (1962) A machine program for theorem-proving, Commun. ACM, 5, (7), 394-397.
[3] A. Biere, M. Heule, H. Maaren, and T. Walsh (2009) Handbook of satisfifiability: Volume 185 frontiers in artifificial intelligence and applications.
[4] A. Leporati, and S. Felloni (2007) Three quantum algorithms to solve 3-sat. Theoretical Computer Science, 372, 218-241.
[5] J. Wang, J. Chen, C. Yu, and L. Wang (2012) A quantum method to test the satisfifiability of boolean functions. 1-5.
[6] L. Grover (1996) Fast quantum mechanical algorithm for database search. Proceedings of the 28th Annual ACM Symposium on the Theory of Computing.
[7] J. C. Garcia-Escartin, P. Chamorro-Posada (2011) Equivalent quantum circuits.
[8] C. Figgatt, D. Maslov, K. Landsman, N. Linke, S. Debnath, and C. Monroe (2017) Complete 3-qubit grover search on a programmable quantum computer. Nature Communications, 8.
[9] Z. Diao, M. Zubairy, and G. Chen (2002) A quantum circuit design for grover's algorithm. Zeitschrift f¨ur Naturforschung, A 57.
[10] D. Fernandes, and I. Dutra (2019) Using grover's search quantum algorithm to solve Boolean satisfiability problems: Part i, XRDS: Crossroads. The ACM Magazine for Students, 26, 64-66.
[11] S.-T. Cheng, and M.-H. Tao (2007) Quantum cooperative search algorithm for 3- sat. Journal of Computer and System Sciences, 73, 123-136.
[12] A. Montanaro (2015) Quantum walk speedup of backtracking algorithms. Theory of Computing, 14.
[13] M. Jarret, and K. Wan (2018) Improved quantum backtracking algorithms using effective resistance estimates. Physical Review, A 97.