A Generic Variable Inputs Quantum Algorithm for 3-SAT Problem

Ping Wang
College of Electronics and Information Engineering, Shenzhen University, Shenzhen, China
wangping@szu.edu.cn

Guangqiang Liu
College of Electronics and Information Engineering, Shenzhen University, Shenzhen, China
2172262931@email.szu.edu.cn

Liyan Liu
College of Electronics and Information Engineering, Shenzhen University, Shenzhen, China
1810262094@email.szu.edu.cn

***Abstract*—The Grover's algorithm can find an item satisfying certain properties in a search space of size *N* with circuit complexity $O(\sqrt{N})$, where the circuit are formed by the *G* operator one by one in series. The huge circuit scale is the main obstacle to the practical application of the Grover's algorithm. In this paper, we present a generic variable inputs quantum algorithm by time-space tradeoff to reduce the scale of the quantum circuit. We combine the proposed algorithm in the paper with the Schöning's algorithm to solve the 3-SAT problem which is a NP-complete problem. Compared with previous algorithms, as the input variables *n* of 3-SAT problem increases, the advantages of our algorithm is obvious.**

Keywords—quantum algorithm, Grover's algorithm, 3-SAT, Schöning's algorithm.

I. INTRODUCTION

Quantum computing is a new field about the physics, mathematics, and intersection of computer science. As a new technology, its main advantage is that it has fast calculation speed. Compared with classic computers under certain problems, its operation speed has been reduced from the exponential time of the classic computer to the polynomial time. It relies on the principles of quantum mechanics to obtain solutions to satisfiability problems. In 1982, Feynman discovered that simulating quantum processes on classical computers requires super-exponential storage space and running time[1], so he began to consider combining quantum mechanics and classical computing, and introduced the concept of quantum computers. In 1994, Shor proposed a large number prime factorization quantum algorithm[2], which proved that the large number prime factorization can be completed in polynomial time by using a quantum computer. Although the algorithm proposed by Shor can reduce the exponential time to the polynomial time of the classical algorithm, the algorithm is only effective for some specific problems. After Shor's pioneering work, Grover proposed a quantum search algorithm in 1996[3], which only provides a quadratic improvement compared with the polynomial acceleration of the Shor algorithm. It can solve a search problem in an unstructured database of size *N* with circuit complexity $O(\sqrt{N})$. The Grove's algorithm has a square acceleration compared to the classic algorithm, which can be used to speedup algorithms for NP-complete problems.

The SAT problem is one of the NP-complete problems, to determine the satisfiability of a given logical formula. The k -SAT problem is a Boolean satisfiability problem, where k represents k variables for each constraint in the satisfiability problem. A special example is that when $k = 3$, which is the first NP-complete problem that was proved by Cook in 1997[4] . When k> 3, k -SAT problem can be reduced to 3-SAT problem[5]. The solution of 3-SAT problem can be defined by a complete set of variable assignments such that every variable has some value, no variable is assigned conflicting values, and all the constraints are satisfied. At present, many algorithms exist to solve the 3-SAT problem. The DPLL algorithm proposed by Davis and PutIl in 1960 is one of the most well-known complete algorithms[6][7]. The research and development of subsequent complete algorithms are mostly based on the DPLL algorithm. Schöning's algorithm[8] is one of the best classical algorithm known for solving the 3-SAT problem, which is the non-deterministic algorithm and applies a local search of $3n$ steps on randomly selected points from the search space. Ambainis proposed the quantum amplitude amplification technique to improve the classic algorithm for solving 3-SAT problem[9]. The improved algorithm provides a squared acceleration over the classic algorithm. In 2007, Alberto Leporati et al. proposed three "quantum" brute force algorithms to solve the 3-SAT problem[10]. In 2008, an improved quantum-inspired evolutionary algorithm to solve 3-SAT problem was presented by Xiaoyue Feng et al.[11]. The algorithm uses a novel quantum revolving gate strategy to adjust the direction of the quantum gate for updating the quantum population.

II. A GENERIC VARIABLE INPUTS QUANTUM ALGORITHM

Grover's algorithm is used to search an item satisfyingcertain properties out of a database of *N* items with running time complexity $O(1)$ and circuit complexity $O(\sqrt{N})$. The practical application of the Grover's algorithm is limited by the huge quantum circuit scale, since quantum computer by current techniques can be equipped with a small number of qubits.

For a function f, which takes x as an input integer, in the range 0 to $2^n - 1$. Let $x' \in \{x'_1, x'_2, \cdots, x'_m\}$ denote the set of the solution of $f(x)$, i.e $f(x) = 1$ if $x = x'$ and $f(x) = 0$ if $x \neq x'$. When the number of input bits n is relatively large, the quantum circuit consumes a lot of resources. To reduce the circuit scale, we can divide the original problem into two parts: n_1 and n_2, where $n = n_1 + n_2$. Correspondingly, we convert the original problem into 2^{n_1} subtasks, and the search space of each subtask is 2^{n_2}, which can be solved by the Grover search algorithm. Therefore, we can find the

solution of $f(x)$ in 2^n elements with running time complexity $O(2^{n_1})$ and circuit complexity $O(2^{n_2/2})$. The circuit framework of this approach is shown by Fig.1. Register 1 contains n_1 bits, which is initialized according to the binary representation of x_1, where $x_1 \in \{0,1,2,\cdots,2^{n_1}-1\}$, and the register 2 contains n_2 qubits initially in the state $|0\rangle^{\otimes n_2}$.

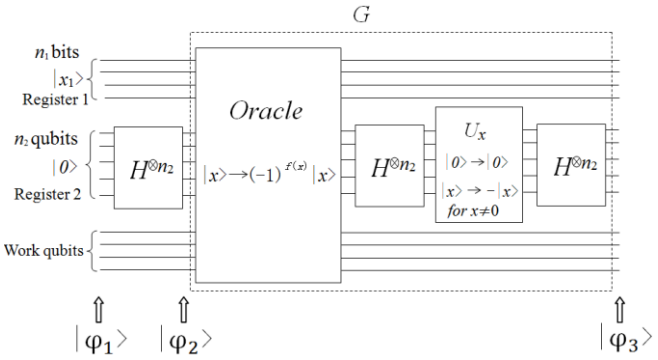


Fig. 1. The circuit framework of the generic variable inputs quantum algorithm.

We need to point out that, for any $x_1 \in \{0,1,2,\cdots,2^{n_1}-1\}$ corresponding to a subset

$$X_1 = \{2^{n_2}x_1 + 0, 2^{n_2}x_1 + 1, \ldots, 2^{n_2}(x_1+1) - 1\}$$

Therefore, the size of each subset is 2^{n_2}.

Let $|\varphi_1\rangle$, $|\varphi_2\rangle$ and $|\varphi_3\rangle$ indicate the state of different phases in the system. Note that, in the following analysis, we ignore the work qubits in Fig.1, which will not affect the effect of the algorithm.

$$|\varphi_1\rangle = |x_1\rangle|0\rangle^{\otimes n_2}$$

Hadamard transform is used to put the computer in the equal super-position state, hence

$$|\varphi_2\rangle = I_{n_1} \otimes H^{\otimes n_2}|\varphi_1\rangle = |x_1\rangle \frac{1}{\sqrt{2^{n_2}}} \sum_{x=0}^{2^{n_2}-1} |x\rangle$$

The operator *Oracle* is used to mark the location of the solution of $f(x)$, so it maps state $|x'\rangle$ to $-|x'\rangle$ and the others keep original states. Generally the operator *Oracle* can be written as $I - 2|x'\rangle\langle x'|$. The operator U_x is a conditional phase shift operator. Its main function is to make all states receive a phase shift of -1, except the state $|0\rangle$, i.e $|x\rangle$ to $-|x\rangle$, if $x \neq 0$. Generally the operator U_x can be written as $2|0\rangle\langle 0| - I$. Let G denote the iteration operator as illustrated in the dotted box of Fig.1.

$$G = (I_{n_1} \otimes H^{\otimes n_2}) U_x (I_{n_1} \otimes H^{\otimes n_2}) Oracle$$

Let $|\varphi_3\rangle$ dentoe the state after r iterations, for each x_1, we have

$$|\varphi_3\rangle = G^r|\varphi_2\rangle = G^r|x_1\rangle \frac{1}{\sqrt{2^{n_2}}} \sum_{x=0}^{2^{n_2}-1} |x\rangle$$

$$= |x_1\rangle \sum_{x=0}^{2^{n_2}-1} [-a_{x,r} + 2\langle a_r\rangle]|x\rangle$$

Where

$$a_{x,r} = -a_{x,r-1} + 2\langle a_{r-1}\rangle$$

$$\langle a_r\rangle = \frac{1}{2^{n_2}} \sum_{x=0}^{2^{n_2}-1} a_{x,r}$$

denote the coefficient of the state $|x\rangle$ and the average of $a_{x,r}$ correspondingly. When $r = 1$,

$$a_{x,1} = \frac{1}{\sqrt{2^{n_2}}}(-1)^{f(x)}$$

Assuming that, for each $x_1 \in \{0,1,2,\cdots,2^{n_1}-1\}$, there is either zero or one solution, from the research results of[12], we need to iterate r of the order is

$$r = \left\lceil \pi / 4\sqrt{2^{n_2}} \right\rceil$$

After r iterations, if current subtask does not contain the solution, i.e $x' \notin X_1$, we have

$$|\varphi_3\rangle = |x_1\rangle \frac{1}{\sqrt{2^{n_2}}} \sum_{x=0}^{2^{n_2}-1} |x\rangle$$

and, if current subtask contains the solution, i.e $x' \in X_1$

$$|\varphi_3\rangle \approx [-a_{x',r} + 2\langle a_r\rangle]|x'\rangle \frac{|0\rangle - |1\rangle}{2} \approx |x'\rangle$$

We will get a measured value, if we measure the register 2 in the system after r iterations. If current subtask constains the solution, we will get the solution with a maximum probability which close to one. Let x'' denote the measurement result of register 2 in the system, we will get the solution

$$x' = 2^{n_2}x_1 + x''$$

If current subtask does not contain the solution, it means that the oracle circuit do nothing, a measurement on the register 2 will yield a random value from $\{0,1,2,\cdots,2^{n_2}-1\}$ with equal probability after repeating r times. Therefore, this cannot accurately find the solution for function $f(x)$.

Generally, the number of solutions of the original search problem is relatively small. Therefore, the current subtask usually does not contain a solution x'. We can apply one more operation $H^{\otimes n_2}$ on register 2 just before final measurement to make the result analysis easier. We omit this

operation and the measurement in Fig.1 for simplicity. In the case where current subtask does not contian the solution, before applying $H^{\otimes n_2}$ to the system in the final step, the state of register 2 is an equal superposition state. Therefore, a measurement will yield 0 for all qubits in the register 2. If current subtask contains the solution, we will get a random value from $\{0,1,2,\cdots,2^{n_2}-1\}$ with equal probability, when we measure the state on the register 2. It can be seen that if the measured value is non-zero, whcih means the current subtask contains a solution, so we just need to execute the circuit once more with current x_1 without the final operation $H^{\otimes n_2}$ on register 2 to find the exact value of the solution x'. Let x'' denote the new measurement result of register 2, in fact, we have got the solution $x' = 2^{n_2}x_1 + x''$. If the measurement result is 0, then take the next value of x_1 and execute this system again, until the value of x_1 is traversed. If $f(x)$ has m solutions, the algorithm needs to be repeated $2^{n_1}/m$ times to find a solution. In general, this algorithm can find the solution of the problem under the condition that the running time complexity is $O(2^{n_1})$ and the circuit complexity is $O(2^{n_2/2})$.

Note that, if the current subtask contains a solution, when we measure register 2 finally , the measurement state will has the $1/2^{n_2}$ probability to collapse to the $|0\rangle$ state, which means that our algorithm has a $1/2^{n_2}$ probability of not finding a solution to the original problem.

III. Quantum search combined with shöning's algorithm for 3-SAT problem

Let $F(X)$ represent a 3-SAT problem and C_i represents constraints. The mathematical formulation of the 3-SAT problem is as follows:

- Let there be a set of n variables $X = \{x_1, x_2, \cdots, x_n\}$ where $x_i \in \{0,1\}$.
- Let there be a Boolean expression $F(X) = C_1 \wedge C_2 \wedge \cdots \wedge C_m$ with $C_2 = l_{j1} \vee l_{j2} \vee l_{j3}$ where l_{jk} is either a variable x_i or its negation $\bar{x_i}$.
- The 3-SAT problem consists in determining the set of values for the variables $X = \{x_1, x_2, \cdots, x_n\}$ that makes $F(X)$ true.

In the 3-SAT problem, there are some common characteristics between different non-solutions. For example, if a particular combination assigned to a subset of variables violates one or more constraints, the assignment containing that particular combination must not be the solution. If the partial combination assignment does not violate the constraints, then it may be a partial solution of the 3-SAT problem. We can get a complete solution of 3-SAT problem by systematically extending partial solution.

The main idea of the algorithm is to search a partial solution space to avoid searching the entire space. Let's divide the n input variables $\{x_1, x_2, \cdots, x_n\}$ of the 3-SAT problem into two parts: n_1 and n_2, where $n_1 + n_2 = n$, and n_1 contains primary variables $\{x_1, x_2, \cdots, x_{n_1}\}$ and n_2 contians variables $\{x_{n_1+1}, x_{n_1+2}, \cdots, x_n\}$. In general, any partial solution of 3-SAT problem can be tested according to partial constraints, that is, only those constraints involving the primary variables n_1 are tested. A partial solution that satisfies all these constraints can be considered as *could-be* solutions. Our algorithm can be speeded up by terminating search along paths that are not descendants of *could-be* solutions, thus avoiding searching the entire space.

The size of the search space of *could-be* solutions (assignment of the primary variable n_1) is 2^{n_1}. Assuming that there are n_1' *could-be* solutions in primary variables n_1, the probability of finding one of them by using a random search is thus $n_1'/2^{n_1}$. Thus, one needs of the order of

$$\alpha = 2^{n_1} / n_1'$$

iterations to find one *could-be* solution. The size of descendants of a *could-be* solution which is obtained by assigning value to the subset of variables n_2 is 2^{n_2}. Therefore, it takes an average of

$$\beta = 2^{n_2}$$

Iterations to pass through the entire space of the descendants. Assuming that, for each *could-be* solution, there is either zero or one solution of 3-SAT problem, on average, the whole procedure needs to be repeated n_1', since we have n_1' *could-be* solutions. Thus, the total number of iterations required to find a solution of an average instance is approximately equal to

$$T(n_1) \simeq n_1'(\alpha + \beta) = 2^{n_1} + n_1' 2^{n_2}$$

Let $p(n_1')$ denote the probability that a partial solution of 3-SAT in the primary variables n_1. We have $p(n_1) = n_1'/2^{n_1}$, therefore

$$T(n_1) \simeq 2^{n_1} + p(n_1)2^n$$

Let's use the generic variable inputs quantum algorithm introduced earlier to solve 3-SAT problem. The input x_1 of the quantum search algorithm denoted a *could-be* solution of 3-SAT problem. To prevent the input x_1 of the quantum search from colliding with the variable name of 3-SAT, we use X_1' to represent x_1 of the quantum search. To optimize the running time of the algorithm, we can find a *could-be* solution X_1' by Schöning's algorithm[8] with running time complexity $O((4/3)^{n_1})$. The main idea of shöning's algorithm is to randomly select a variable in a constraint that not be satisfied by the current assignment, and flip its value. Schöning, by introducing constraints to bound the possibility of finding a solution for a given random walk, limits the problem of candidate states that tend to move away from the solution. In general, the average search time of the whole process of quantum search combined with shöning's algorithm to find the a solution of 3-SAT problem is approximately equal to

$$T(n_1) \simeq (4/3)^{n_1} + p(n_1)2^{(n+n_1)/2}$$

When $n \to \infty$, according to the research results by Cerf et al. [13], we have

$$p(n_1) \simeq 2^{-n(\lambda/\lambda_c)(n_1/n)^3}$$

where $\lambda = \xi/\mu$ measures the difficulty of the problem and $\lambda_c = 2^3 log(2)$ is the critical value around which the problem is the most difficult, where ξ is the number of constraints. Now, let $\lambda = \lambda_c$, i.e, when the difficulty of the problem is maximum for a given size n. Substituting the above formula into $T(n_1)$, we have

$$T(n_1) \simeq (4/3)^{n_1} + 2^{(n+n_1)/2 - n_1^3/n^2}$$

Let $i = n_1/n$, which represents the ratio of the primary variables to all variables.

$$T(i) \simeq (4/3)^{in} + 2^{n((1+i)/2 - i^3)}$$

we want to find the value of i that minimizes thecomputation time $T(i)$, so we can calculate i by taking the derivative of the function $T(i)$.

$$T'(i) \simeq (4/3)^{in} ln(4/3) n + 2^{n((1+i)/2 - i^3)} ln(2) n(1/2 - 3i^2)$$

The quantum search combined with shöning's algorithm for 3-SAT problem is described as Fig.2.

Quantum search combined with shöning's algorithm:

1. Select an initial assignment τ ,where $\tau \in \{0,1\}^{n_1}$, uniformly at random and set a counter $c = 1$.
2. Rerpeat $3n_1$:
   1) $c = c + 1$;
   2) If all the constraints which contain primary variables n_1 are satisfied by the current assignment: stop and goto 4.
   3) Let C' be some constraint which contains the primary variables n_1 not being satisfied by the current assignment.
   4) Select one of its three variables at random from C', and flip its value.
3. If $c \geq 3n_1$ then goto 1.
4. Quantum search algorithm $(\tau, |0\rangle^{\otimes n_2})$
5. Measure register 2 and get ${x_2}'$:
   If ${x_2}' = 0$
   goto 1.
   Else
   Repeat stept 4,5 without the final operation $H^{\otimes n_2}$ on register 2, output $X = 2^{n_2}\tau + {x_2}'$.
   Stopt.

Fig. 2. The procedure of the quantum search combined with shöning's algorithm for 3-SAT problem.

The plot of the function $T'(i)$ is shown as Fig.3, where the variables $n = 25$. We can get the minimum value of $T(i)$ from Figure 3, where $i \approx 0.86$. Let $T'(i)$=0, when $n \to \infty$, $i \approx 0.83$, which minimizes the computation time $T(i)$.

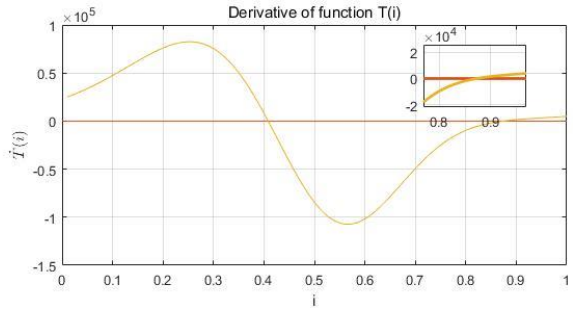


Fig. 3. The derivative of the function $T(i)$.

Table I is a comparison of the quantum search combined with shöning's algorithm, shöning's algorithm and Grover's algorithm. Let "Q with S" denote our algorithm.

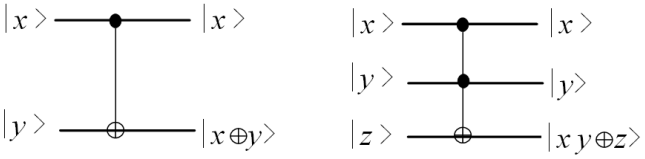
TABLE I. COMPARISON OF COMPLEXITY OF DIFFERENT ALGORITHMS

n / algorithm	10	20	30	40	50
Q with S	17	194	2124	23235	251892
Shöning	17	315	5599	99437	1765780
Grover	32	1024	32768	1048576	33554432

It is clear that when n is larger, the advantage of our algorithm is more obvious.

IV. DESIGN OF ORACLE OF QUANTUM CIRCUITS FOR 3-SAT

In a classical quantum gate, a controlled-NOT gate can make the state $|x, y\rangle \to |x, x \oplus y\rangle$, where $|x\rangle$ is the control bit[14]. If $|x\rangle = |0\rangle$, then $|y\rangle$ will be the same as the input. If $|x\rangle = |1\rangle$, then $|y\rangle$=$|\bar{y}\rangle$. The circuit of the controlled-NOT gate is shown as *a*) in Fig.4. Toffoli gate is similar to the controlled-NOT gate, but with two controlling bits $|x\rangle$ and $|y\rangle$, it can make the state $|z\rangle \to |xy \oplus z\rangle$, as *b*) in Fig.4.



a) *controlled-NOT gate* *b*) *Toffoli gate*

Fig. 4. Controlled-NOT gate and Toffoli gate.

We can design a quantum gate with n control bits through some Toffili gates. For simplicity, we only give a simple frame diagram with n control bits, as in Fig.5.

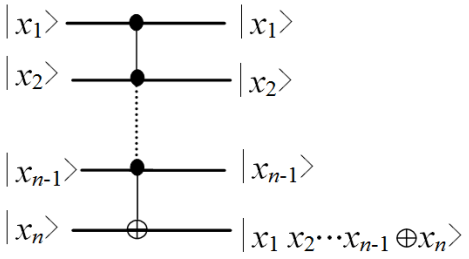


Fig. 5. *n* bits control gate.

Any 3-SAT problem can be represented as *XOR* form. Let $F(X)$ is a boolean expression of 3-SAT problem, For

example

$$F(X) = (x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee \overline{x_3})$$

The logical symbol ∧ is replaced by the arithmetical operator product(·) and thesymbol ∨ is replaced by the arithmetical operator plus(+). We have

$$\begin{aligned} F(X) &= (x_1 + x_2 + x_3)(x_1 + \overline{x_2} + x_3)(x_1 + \overline{x_2} + \overline{x_3}) \\ &= x_1 + \overline{x_2}x_3 = x_1 \oplus \overline{x_2}x_3 \oplus x_1\overline{x_2}x_3 \end{aligned}$$

Therefore, we can design the circuit of the *Oracle* for 3-SAT promble, as in Fig.6.

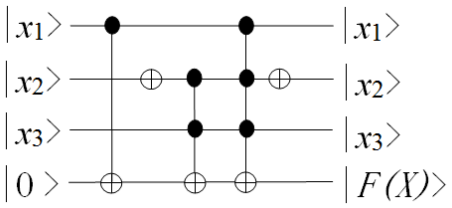


Fig. 6. The circuit of the Oracle for 3-SAT

V. CONCLUSIONS

It is well known that Grover's algorithm can find a solution in a search space of size N with quantum circuit complexity $O(\sqrt{N})$. The huge circuit scale is the main obstacle to the practical application of the Grover's algorithm since quantum computer by current techniques can be equipped with only a few qubits. We propose a generic variables inputs for Grover's algorithm to reduce the circuit scale by time-space tradeoff. According to some common features between different non-solutions in 3-SAT problem, we apply a generic variables inputs for Grover's algorithm combined with shöning's algorithm for sloving 3-SAT problem. When the variables n of the 3-SAT problem is larger, the advantage of our algorithm is more obvious.

ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China (No. 61872245), Shenzhen Fundamental Research fund (No. JCYJ20170818144026871, JCYJ20180305123639326).

REFERENCES

[1] Feynman, Richard P. "Quantum Mechanical Computers." Foundations of Physics, vol. 16, no. 6, 1986, pp. 507–531.

[2] Shor, Peter W. "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer." Siam Review, vol. 41, no. 2, 1999, pp. 303–332.

[3] Grover, Lov K. "A Fast Quantum Mechanical Algorithm for Database Search." Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing, 1996, pp. 212–219.

[4] Cook, Stephen A. "The Complexity of Theorem-Proving Procedures." Proceedings of the Third Annual ACM Symposium on Theory of Computing, 1971, pp. 151–158.

[5] Hartmanis, and Juris. "Computers and Intractability: A Guide to the Theory of NP-Completeness (Michael R. Garey and David S. Johnson)." Siam Review, vol. 24, no. 1, 1982, pp.90-91.

[6] Davis, M. "A machine program for theorem-proving." Communications of the ACM, vol. 5, no. 7, 1962, pp. 394-397.

[7] Davis, Martin, and Hilary Putnam. "A Computing Procedure for Quantification Theory." Journal of the ACM, vol. 7, no. 3, 1960, pp. 201–215.

[8] Schöning, T. "A Probabilistic Algorithm for k-SAT and Constraint Satisfaction Problems." 40th Annual Symposium on Foundations of Computer Science (Cat. No.99CB37039), 1999, pp. 410–414.

[9] Ambainis, A. "Quantum Search Algorithms." Sigact News, vol. 35, no. 2, 2004, pp. 22–35.

[10] Leporati, Alberto, and Sara Felloni. "Three Quantum Algorithms to Solve 3-SAT." Theoretical Computer Science, vol. 372, no. 2, 2007, pp. 218–241.

[11] Feng, Xiaoyue, et al. "Improved Quantum-Inspired Evolutionary Algorithm and Its Application to 3-SAT Problems." 2008 International Conference on Computer Science and Software Engineering, vol. 1, 2008, pp. 333–336.

[12] Grover, Lov K. "Quantum Mechanics Helps in Searching for a Needle in a Haystack." Physical Review Letters, vol. 79, no. 2, 1997, pp. 325–328.

[13] Cerf, Nicolas J., et al. "Nested Quantum Search and NP-Hard Problems." Applicable Algebra in Engineering, Communication and Computing, vol. 10, no. 4, 2000, pp. 311–338.

[14] Yanofsky, Noson S., and Mirco, Mannucci A. Quantum computing for computer scientists. Cambridge University Press, 2008.

[15] M. Saravanan and A. Priya (2019). An Algorithm for Security Enhancement in Image Transmission Using Steganography. Journal of the Institute of Electronics and Computer, 1, 1-8.

[16] G. H. Rosa, J. P. Papa (2019). Soft-Tempering Deep Belief Networks Parameters Through Genetic Programming. Journal of Artificial Intelligence and Systems, 1, 43–59.