Quantum solution to the Boolean satisfiability problem

Yash Prabhat[1,*]

[1]Department of Physics, Indian Institute of Science Education and Research, Bhopal 462066, India

(Dated: June 14, 2025)

The Boolean satisfiability problem (SAT) is central to computational complexity and has wide-ranging applications. We introduce a quantum algorithm that deterministically identifies satisfying assignments by leveraging quantum operators to efficiently remove non-satisfying states from a uniform superposition. Our method constructs a quantum circuit in polynomial time, with classical complexity $\mathcal{O}(m^2 \cdot n^3)$ and worst-case quantum circuit depth $\mathcal{O}(m^2)$, where m is the number of clauses and n the number of variables. This represents a significant departure from previous quantum and classical methods, which generally require exponential resources in the worst case.

I. INTRODUCTION

The Boolean satisfiability problem (SAT) is a fundamental challenge in computational complexity theory, recognized as the first NP-complete problem. Determining whether a propositional formula possesses a satisfying assignment of its variables has profound implications for automated theorem proving, hardware verification, and artificial intelligence [1]. While classical algorithms can solve 2-SAT in polynomial time through implication graph analysis, the general k-SAT problem for $k \geq 3$ is NP-hard. Best-known classical solvers such as DPLL and its modern variants, including conflict-driven clause learning (CDCL), require exponential time in the worst case, with complexity scaling as $\mathcal{O}(2^n)$ for n variables [2]. Stochastic local search methods like WalkSAT offer practical speed on industrial instances but trade completeness for efficiency and remain exponential in the worst case [3, 4].

The emergence of quantum computing has introduced new paradigms for attacking SAT, leveraging quantum parallelism and interference to potentially outperform classical methods. Grover's algorithm, for instance, achieves a quadratic speedup over classical unstructured search, resulting in $\mathcal{O}(\sqrt{2^n})$ query complexity [5]. While this represents a significant theoretical improvement, it remains exponential in n and does not exploit the structure inherent in many SAT instances. Recent advances in quantum walk formulations and the Quantum Approximate Optimization Algorithm (QAOA) have demonstrated improved clause satisfaction probabilities, but these methods struggle with hierarchical variable assignment verification and are limited by circuit depth and training difficulties at critical problem densities [6, 7].

In this work, we present a novel quantum approach that leverages the ability of quantum operators to remove non-satisfying states from a uniform superposition in a single step. This leads to polynomial-time circuit construction and opens the door to practical applications of SAT solvers at industrial scale [4]. Classically, removing N non-satisfying entries would require N queries, but our quantum approach accomplishes this in a single operation, thus realizing a clear quantum advantage. The state removal process is inspired by the encoding methods for our work on structured quantum search [8]. The classical complexity of generating the quantum circuit is $\mathcal{O}(m^2 n^3)$, where m is the number of clauses and n is the number of literals, while the worst-case quantum circuit depth is $\mathcal{O}(m^2)$ [9].

The achievement of polynomial complexity in SAT solving marks a paradigm shift for this fundamental problem. Classical and quantum approaches have historically faced an insurmountable exponential barrier for $k \geq 3$, but our method demonstrates that, with careful exploitation of quantum parallelism and structured state removal, it is possible to design algorithms whose resource requirements scale polynomially with problem size. This breakthrough opens the door to the practical application of SAT solvers to industrial-scale instances that were previously considered intractable, heralding a new era in computational problem-solving for NP-complete problems [4, 9].

II. METHODS

A. Boolean satisfiability problem (SAT)

The Boolean satisfiability problem (SAT) is a foundational decision problem in computational complexity and mathematical logic. Given a Boolean formula composed of variables and logical operators, SAT asks whether there exists an assignment of truth values (TRUE/FALSE) to the variables that make the entire formula evaluate to TRUE. This problem serves as a cornerstone for understanding NP-completeness and has profound implications across computer science and quantum computing. A Boolean formula ϕ in conjunctive normal form (CNF) is expressed as:

$$\phi = \bigwedge_{i=1}^{m} C_i \quad \text{where each clause } C_i = \bigvee_{j=1}^{k} l_{ij} \qquad (1)$$

Here, l_{ij} represents a literal (a variable x_p or its negation $\bar{x}_p$, $p \in \{0, \ldots, n-1\}$). The problem is called k-SAT if

* Contact author; yashp20@iiserb.ac.in

every clause C_i contains exactly k literals. For $k \geq 3$, k-SAT is NP-complete, while 2-SAT is solvable in polynomial time. For simplicity, in this work, we shall use the notation of the variable x_j for the literal l_{ij} and their respective data qubit as q_j, $j \in \{1, \ldots, n\}$.

B. The idea

Our approach is based on the fact that a single quantum operator is capable of removing all the states that would not satisfy a particular clause from an equal superposition of all states. It is inspired by the dataset encoding method proposed in [8] for the structured search algorithm. Classically, to remove any arbitrary N entries that would not satisfy a clause would require N queries, hence the quantum advantage. The operator in question is based on the Hadamard operator $\hat{\mathrm{H}}$ defined as:

$$\hat{\mathrm{H}} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}. \tag{2}$$

The action of $\hat{\mathrm{H}}$ on:

$$\begin{aligned} \hat{\mathrm{H}} \left|0\right\rangle = \left|+\right\rangle, \hat{\mathrm{H}} \left|+\right\rangle = \left|0\right\rangle \\ \hat{\mathrm{H}} \left|1\right\rangle = \left|-\right\rangle, \hat{\mathrm{H}} \left|-\right\rangle = \left|1\right\rangle \end{aligned} \tag{3}$$

We only wish to work with $\{0, 1, +\}$ states so we add the Pauli operator $\hat{\mathrm{Z}}$ defined as

$$\hat{\mathrm{Z}} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}. \tag{4}$$

The Eq. (3) can be modified as

$$\begin{aligned} \hat{\mathrm{H}} \left|0\right\rangle = \left|+\right\rangle, \hat{\mathrm{H}} \left|+\right\rangle = \left|0\right\rangle \\ \hat{\mathrm{Z}} \cdot \hat{\mathrm{H}} \left|1\right\rangle = \left|+\right\rangle, \hat{\mathrm{H}} \cdot \hat{\mathrm{Z}} \left|+\right\rangle = \left|1\right\rangle \end{aligned} \tag{5}$$

Consider a single literal clause C_i as $C_i = l_{ij}$, where $l_{ij} = x_j$ or $l_{ij} = \bar{x}_j$ for $j \in \{1, \ldots, n\}$, where n is the total number of literals. We start with an equal superposition of all 2^n states $\left|\psi\right\rangle$ as

$$\left|\psi\right\rangle = \hat{\mathrm{H}}^{\otimes n} \left|0\right\rangle^{\otimes n} = \left|+\right\rangle^{\otimes n} = \left|+\right\rangle_1 \left|+\right\rangle_2 \ldots \left|+\right\rangle_j \ldots \left|+\right\rangle_n, \tag{6}$$

where we have n data qubits q_j that process all n literals in parallel. Here, we wish to remove the states that dissatisfy clause C_i, we achieve this via the operation

$$\hat{\mathrm{I}}^{\otimes(j-1)} \otimes \hat{\mathrm{H}} \cdot \hat{\mathrm{Z}} \otimes \hat{\mathrm{I}}^{\otimes(n-j)} \left|\psi\right\rangle = \left|+\right\rangle_1 \left|+\right\rangle_2 \ldots \left|1\right\rangle_j \ldots \left|+\right\rangle_n, \quad \text{if } l_{ij} = x_j, \tag{7}$$

$$\hat{\mathrm{I}}^{\otimes(j-1)} \otimes \hat{\mathrm{H}} \otimes \hat{\mathrm{I}}^{\otimes(n-j)} \left|\psi\right\rangle = \left|+\right\rangle_1 \left|+\right\rangle_2 \ldots \left|0\right\rangle_j \ldots \left|+\right\rangle_n, \quad \text{if } l_{ij} = \bar{x}_j, \tag{8}$$

where $\hat{\mathrm{I}}$ is the identity operator

$$\hat{\mathrm{I}} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{9}$$

Here, we remove 2^{n-1} states that would not satisfy the clause C_i with a single operation. However, we can not use the operations from Eq. (5) for multiple clauses as they are unitary and thus cannot form an AND gate. To overcome this we devise an annihilation operation $\hat{\mathrm{A}}$ to remove non-satisfying states from the superposition using ancilla in state $\left|0_a\right\rangle$ qubits and controlled not (CX) operations.

$$\hat{\mathrm{A}} = \mathrm{CX}_{j,a} \cdot \mathrm{CX}_{a,j} \tag{10}$$

where the unitary operations for $\mathrm{CX}_{j,a}$ and $\mathrm{CX}_{a,j}$ are

$$\mathrm{CX}_{j,a} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \text{ and } \mathrm{CX}_{a,j} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}. \tag{11}$$

$\hat{\mathrm{A}}$ acts on one data qubit q_j and ancilla a, set to state $\left|0_a\right\rangle$. The quantum circuit for the operator $\hat{\mathrm{A}}$ is given in Fig. 1. The action of $\hat{\mathrm{A}}$ on respective qubits in the concerned states can be written as

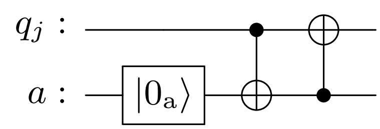

FIG. 1. Quantum circuit for operator $\hat{\mathrm{A}}$.

$$\begin{aligned} \hat{\mathrm{A}} \left|0_a\right\rangle \left|0\right\rangle_j &= \left|0_a\right\rangle \left|0\right\rangle_j \\ \hat{\mathrm{A}} \left|0_a\right\rangle \left|1\right\rangle_j &= \left|1_a\right\rangle \left|0\right\rangle_j \\ \hat{\mathrm{A}} \left|0_a\right\rangle \left|+\right\rangle_j &= \left|+_a\right\rangle \left|0\right\rangle_j. \end{aligned} \tag{12}$$

Thus, $\hat{\mathrm{A}}$ can be used directly when $l_{ij} = x_j$. We modify the above equation using $\hat{\mathrm{X}}$ operator for the case when $l_{ij} = \bar{x}_j$ as:

$$\begin{aligned} \hat{\mathrm{I}} \otimes \hat{\mathrm{X}} \cdot \hat{\mathrm{A}} \cdot \hat{\mathrm{I}} \otimes \hat{\mathrm{X}} \left|0_a\right\rangle \left|0\right\rangle_j &= \left|1_a\right\rangle \left|1\right\rangle_j \\ \hat{\mathrm{I}} \otimes \hat{\mathrm{X}} \cdot \hat{\mathrm{A}} \cdot \hat{\mathrm{I}} \otimes \hat{\mathrm{X}} \left|0_a\right\rangle \left|1\right\rangle_j &= \left|0_a\right\rangle \left|1\right\rangle_j \\ \hat{\mathrm{I}} \otimes \hat{\mathrm{X}} \cdot \hat{\mathrm{A}} \cdot \hat{\mathrm{I}} \otimes \hat{\mathrm{X}} \left|0_a\right\rangle \left|+\right\rangle_j &= \left|+_a\right\rangle \left|1\right\rangle_j, \end{aligned} \tag{13}$$

where $\hat{\mathrm{X}}$ is the Pauli X operator defined as:

$$\hat{\mathrm{X}} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}. \tag{14}$$

Note that in all the final states of the above two equations, the ancilla qubit is separable from the data qubit;

thus, we can reset and reuse the same ancilla qubit for multiple clauses. We define the extrapolated CX gate in $\hat{\mathrm{A}}$ for $k+1$ qubits with first k qubits as control as CX^{k+1},

$$\mathrm{CX}^{k+1} = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & & \vdots & \vdots \\ 0 & 0 & & 1 & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 & 1 \\ 0 & 0 & \cdots & 0 & 1 & 0 \end{bmatrix}_{2^{k+1} \cdot 2^{k+1}} \tag{15}$$

We define an operator $\hat{\mathrm{C}}_i$ that would remove the states which do not satisfy the clause C_i as

$$\begin{aligned} \hat{\mathrm{C}}_i = \bigotimes_{j=1}^{k} \hat{\mathrm{L}}_j \otimes \hat{\mathrm{I}} \cdot \mathrm{CX}^{k+1} \cdot \bigotimes_{j=1}^{k} \hat{\mathrm{L}}_j \otimes \hat{\mathrm{I}} \cdot \\ \hat{\mathrm{I}}^{\otimes \mathrm{t}-1} \otimes \mathrm{CX}_{a,t} \otimes \hat{\mathrm{I}}^{\otimes \mathrm{k}-\mathrm{t}}, \end{aligned} \tag{16}$$

where $\hat{\mathrm{L}}_j$ is

$$\hat{\mathrm{L}}_j = \begin{cases} \hat{\mathrm{I}} & \text{if } l_{ij} = \bar{x}_j \\ \hat{\mathrm{X}} & \text{if } l_{ij} = x_j . \end{cases} \tag{17}$$

Here, t denotes the index of the target qubit q_t for the operation $\mathrm{CX}_{a,t}$. We choose in a manner which least affects the states from the other clauses, this is later discussed in detail in algorithm 1. The quantum circuit for the operation $\hat{\mathrm{C}}_i$ is given in Fig. 2. Here we have taken the first three literals with C_i as $\hat{\mathrm{C}}_i = (l_{i1}, l_{i2}, l_{i3})$.

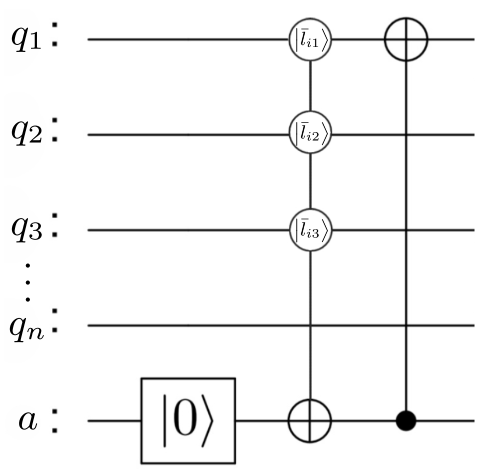

FIG. 2. The quantum circuit for the operation $\hat{\mathrm{C}}_i$. We have taken the first three literals with C_i as $\hat{\mathrm{C}}_i = (l_{i1}, l_{i2}, l_{i3})$. The notation of $|\bar{l}_{ij}\rangle$ in the circle denotes the control state as per the values of literal l_{ij}.

As an example consider C_i, $C_i = (\bar{x}_1, x_2, \bar{x}_3)$. The assignment with values $(x_1 = 1, x_2 = 0, x_3 = 1)$ would not satisfy C_i. Thus, all such states shall be removed by the operator $\hat{\mathrm{C}}_i$ as

$$\begin{aligned} &\hat{\mathrm{C}}_i \left|0_a\right\rangle \left|\psi\right\rangle_n \cdots \left|\psi\right\rangle_4 \left|0\right\rangle_3 \left|0\right\rangle_2 \left|0\right\rangle_1 \\ &\quad = \left|0_a\right\rangle \left|\psi\right\rangle_n \cdots \left|\psi\right\rangle_4 \left|0\right\rangle_3 \left|0\right\rangle_2 \left|0\right\rangle_1 \\ &\qquad \vdots \\ &\hat{\mathrm{C}}_i \left|0_a\right\rangle \left|\psi\right\rangle_n \cdots \left|\psi\right\rangle_3 \left|1\right\rangle_2 \left|0\right\rangle_1 \left|1\right\rangle_1 \\ &\quad = \left|1_a\right\rangle \left|\psi\right\rangle_n \cdots \left|\psi\right\rangle_4 \left|1\right\rangle_3 \left|0\right\rangle_2 \left|0\right\rangle_1 \\ &\qquad \vdots \\ &\hat{\mathrm{C}}_i \left|0_a\right\rangle \left|\psi\right\rangle_n \cdots \left|\psi\right\rangle_4 \left|1\right\rangle_3 \left|1\right\rangle_2 \left|1\right\rangle_1 \\ &\quad = \left|0_a\right\rangle \left|\psi\right\rangle_n \cdots \left|\psi\right\rangle_4 \left|1\right\rangle_3 \left|1\right\rangle_2 \left|1\right\rangle_1 , \end{aligned} \tag{18}$$

where $|\psi_j\rangle$ represents the state of the data qubit q_j, $j \in \{1, \dots, n\}$. Here, the states $\bigotimes_{i=4}^{n} |\psi_i\rangle \otimes |1\rangle_3 |0\rangle_2 |1\rangle_1$ have changed to $\bigotimes_{i=4}^{n} |\psi_i\rangle \otimes |1\rangle_3 |0\rangle_2 |0\rangle_1$, which satisfies the clause C_i. Additionally, all the other states have not been affected. Thus, multiple clauses can be combined in such a manner to solve the Boolean satisfiability problem.

Coming to the problem of the choice of the target qubit, if all the clauses have their separate target qubit without interfering with other clauses, we easily reach the perfect solution by applying them all in sequence. However, if we have an overlap in the literals of two clauses, for example, consider C_i and C_r on literals l_{ij} and l_{rj}, where $l_{ij} = \bar{l}_{rj}$. Here, if we choose q_j as the target qubit for clause C_r, we might create some states that would not satisfy clause C_i. These states need to be removed again with the operator $\hat{\mathrm{C}}_i$ using a different target qubit.

For this purpose, we propose a conjecture that if there exists a state that satisfies all the clauses, it shall remain unaffected by the operations of $\hat{\mathrm{C}}_i$ for $1 \leq i \leq m$. This conjecture can be proved by extrapolating Eq. (18) and using the principle of mathematic induction. Thus, we apply the same clause $\hat{\mathrm{C}}_i$ on multiple target qubits to completely weed out the accidentally created non-solution states again and again. This makes sure we are only left with the solution states.

We propose an additional conjecture that says if there exists a solution state for a Boolean problem, we shall always be able to reach it with infinite random applications of operators $\hat{\mathrm{C}}_i$, $1 \leq i \leq m$. Here, the choice of target qubit q_t, $1 \leq t \leq n$ is also randomized. Thus we suggest that an infinite loop of intersecting clauses would mean no satisfiable assignment to the said Boolean problem exists.

However, we see that by diligently choosing the target qubits we can reach the solution assignment of a Boolean function with a probability of 1 in a circuit depth of only $\mathcal{O}(m^2)$. The complete proof of this conjecture would mean any NP-complete problem can be solved in polynomial time using quantum computers [4]. This work opens the door to a solution of the P=NP problem. However, rigorous mathematical proof is a subject of ongoing research. The example below and the algorithm proposed in the next section can be seen as proof-of-concept

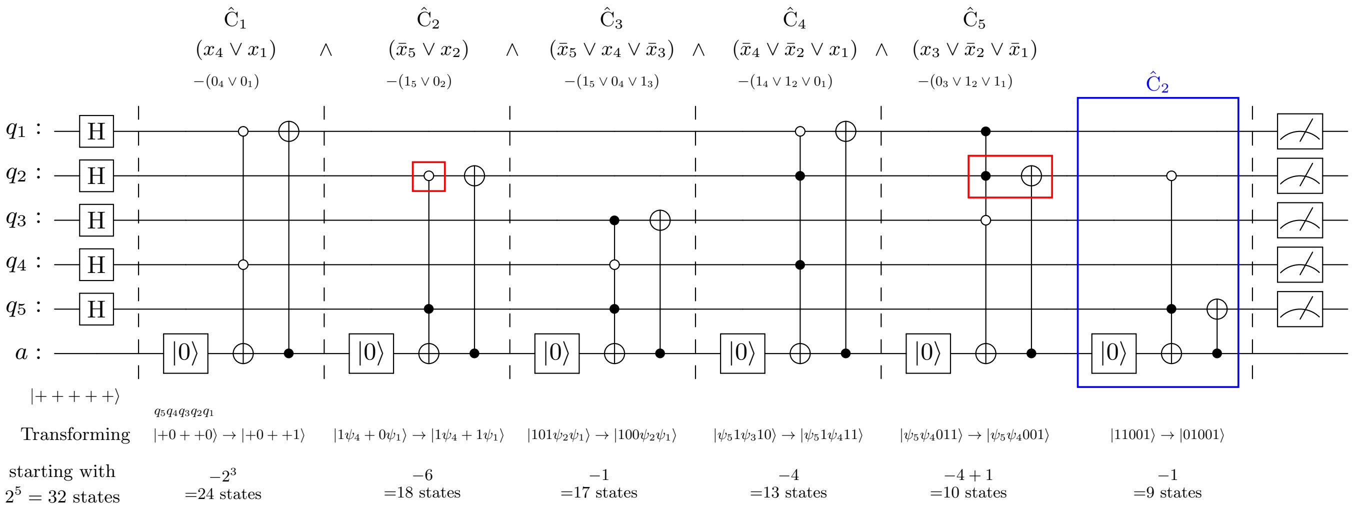

FIG. 3. Quantum circuit to solve the Boolean function in Eq. (19). Running this circuit on a noiseless system would provide a satisfying assignment to the Boolean function with the probability of one. The barrier indicates the circuit for the individual clauses. The text at the bottom indicates how the states are affected by the circuit of the $\hat{\mathrm{C}}_i$ operator. The boxes in red indicate the intersection between clauses C_2 and C_5. Operator $\hat{\mathrm{C}}_5$ has created the state $|11001\rangle$ which does not satisfy clause C_2, thus operator $\hat{\mathrm{C}}_2$ is applied again (blue box).

of such an approach. We have achieved this by minimizing the overlaps caused by the choice of target qubits.

For example, consider a Boolean function

$$(x_4 \vee x_1) \wedge (\bar{x}_5 \vee x_2) \wedge (\bar{x}_5 \vee x_4 \vee \bar{x}_3) \wedge (\bar{x}_4 \vee \bar{x}_2 \vee x_1) \wedge (x_3 \vee \bar{x}_2 \vee \bar{x}_1). \tag{19}$$

Figure 3 shows the quantum circuit to solve the above Boolean function. Here, we have taken the target qubits for the operations $\hat{\mathrm{C}}_i$ in a manner so as to not create the states removed by the previous clauses. Figure 4 shows the states and their counts measured after we run this quantum circuit. All the measured states are our solution states and would satisfy the Boolean in the Eq. (19).

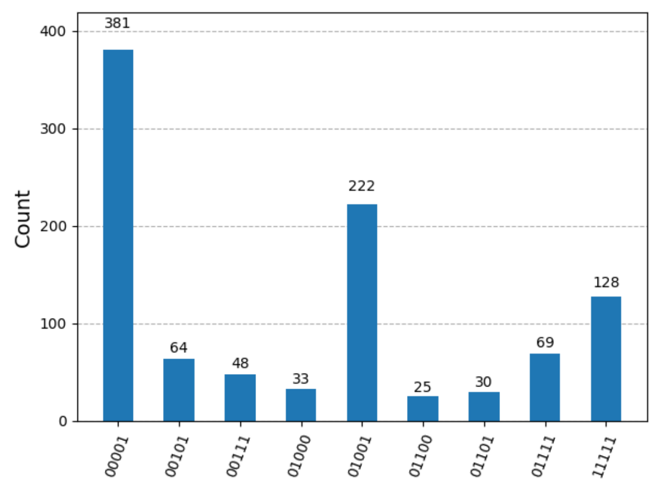

FIG. 4. Measured counts for 1000 shots for the quantum circuit in Fig. 3 [9, 10]. We have achieved a satisfying assignment with a probability of one. All the measured states above are the solution for the Boolean function in Eq. (19); however, they may have different probabilities.

C. The algorithm

Algorithm 1 goes through all possible literals of a clause and finds the target qubit to apply the particular clause with the least circuit depth. The algorithm repeats this for all clauses. When we remove the states that do not satisfy a particular clause by the action of CNOT on a target qubit q_t, we add some undesired states that may not satisfy previous clauses. These states have to be removed again.

We use the FIND function given in algorithm 2 to find all the possible clauses that have to be rechecked due to the choice of a particular target qubit. We achieve this by checking if the target qubit q_t for clause C_i was required to be in the opposite state by a different clause C_r. If so, we apply the operation $\hat{\mathrm{C}}_r$ by using a target qubit q_{t_r} other than q_t. The priority of the choice of the target qubit is given in the algorithm 3. The FIND function has to be repeated again for the new target qubit q_{t_r}. Thus it is a recursive function. We ensure that the target qubit q_t that has once been changed to state $|l_{it}\rangle$ shall not be again acted on to reach state $|\bar{l}_{it}\rangle$. Thus the recursion depth of the circuit can only be n, for n possible literals.

Algorithm 1 $k-$SAT Solution

$m \leftarrow$ number of clauses.
$n \leftarrow$ total number of literals.
Create an equal superposition state $|\psi\rangle$ of all n qubits by $\mathrm{H}^{\otimes n}$ operation.

$$|\psi\rangle = \mathrm{H}^{\otimes n} |0\rangle^{\otimes n}$$

for each integer i following $1 \leq i \leq m$ **do**
  ancilla $a \leftarrow |0\rangle$ {Reset the ancilla qubit}
  $\mathcal{Q} \leftarrow []$
  for all literals l_{ij} in C_i **do**
    $Q \leftarrow []$
    $\mathcal{Q} \leftarrow \mathcal{Q} + [\mathrm{FIND}(C_i, l_{ij}, \mathcal{B}, Q, n)][0]$ {finds intersections with other clauses which intersect with C_i.}
  end for
  $Q \leftarrow \mathrm{min\_len}(\mathcal{Q})$ {minimum length Q in $\mathcal{Q}$; Q contains both q_t and the clauses.}
  for Clauses C_r and target qubits q_t in Q **do**
    Apply $\hat{\mathrm{C}}_r$ on k, a qubits.
    Apply $\mathrm{CX_{a,t}}$ gate to remove the undesired states.
  end for
end for
Measure all n qubits. {measure the states satisfying the Boolean formula.}

Algorithm 2 FIND

Require: $C_i, l_{ij}, \mathcal{B}, Q, n, \mathcal{L}_i = [], \mathcal{C} = [C]$
Ensure: Updated Q, $\mathcal{C}$
if $\mathcal{L}_i = []$ **then**
  $\mathcal{L}_i \leftarrow [l_{ij}]$
end if
if $|\mathcal{L}_i| > n$ **then**
  return $Q, \mathcal{C}$
end if
for $C_r \in \mathcal{B}$ **do**
  for l_{rs} in C_r **do**
    if $l_{rs} = \bar{l}_{ij}$ **then**
      $q_t \leftarrow \textsc{Intersection}(C_r, \mathcal{L}_i, \mathcal{C})$
      if $q_t =$ `null` **then**
        return No Solution
      end if
      if $[q_t, C_r] \in Q$ **then**
        Remove $[q_t, C_r]$ from Q.
      end if
      $\mathcal{C} \leftarrow \mathcal{C} \cup \{C_r\}$
      $Q \leftarrow Q + [q_t, C_r]$
      if $l_{rt} \notin \mathcal{L}_i$ **then**
        $\mathcal{L}_i \leftarrow \mathcal{L}_i \cup \{l_{rt}\}$
        $Q, \mathcal{C} \leftarrow \textsc{Find}(C_r, l_{rt}, \mathcal{B}, Q, n, \mathcal{L}_i, \mathcal{C})$
      end if
    end if
  end for
end for
return $Q, \mathcal{C}$

Here, our goal is to have all the target qubits directed towards our solution states. If we cannot find a suitable target qubit, it would mean that the algorithm has failed to converge, and thus no solution for the Boolean function exists. Exception, currently, the algorithm is unsure how to treat clauses with single literal Booleans, as those clauses can only have a single target qubit. Here, the algorithm fails as it can not find any alternative target qubit to a single literal Boolean function. We have provided a combined flowchart of the algorithms in Fig. (5). Note that for a general Boolean function, currently the algorithm requires the clauses to be sorted in ascending order of the number of literals. These are current limitations, and we are actively working to resolve them in future iterations.

Algorithm 3 Intersection

Require: C_r, $\mathcal{L}_i$, $\mathcal{C}$
for $l_{rj} \in C_r$ **do**
  if $l_{rj} \in \mathcal{L}_i \wedge C_r \notin \mathcal{C}$ **then**
    return q_j
  end if
end for
if $C_r \in \mathcal{C}$ **then**
  for $l_{rj} \in C_r$ **do**
    if l_{rj} has not been applied to qubit q_j **then**
      return q_j
    end if
  end for
end if
for $l_{rj} \in C_r$ **do**
  if $q_j \notin |\mathcal{L}_i|$ **then**
    return q_j
  end if
end for
return `null` {No assignment satisfies}

D. Complexity

The combined complexity of the proposed three algorithms below to produce a quantum circuit is $\mathcal{O}(m^2 \cdot n^3)$, where m is the number of clauses and n is the number of literals in the Boolean function. This is due to two alternate loops, each of length m and n in the algorithm 1 and 2 and the maximum recursive depth of n on the conditional recursive find function, $\mathcal{O}(m \cdot n \cdot m \cdot n \cdot n) = \mathcal{O}(m^2 \cdot n^3)$. The achieved quantum circuits can provide a satisfying assignment of the Boolean satisfiability problem with the probability of one. The worst-case circuit depth of the quantum circuit generated comes out to be $\mathcal{O}(m^2)$. This is because, each clause should not require to have any of the other clauses checked more than once, given the right choice of target qubit. Here, we count each $\hat{\mathrm{C}}_i$ and respective CX operation as a single unit of circuit depth.

We have proposed the said algorithm as a proof of concept for a viable solution to the Boolean satisfiability problem. Currently, the proposed algorithm has some redundancy in the code, along with some corner cases that need to be ironed out. These cases can be resolved by reordering the clauses to avoid getting stuck with a wrong

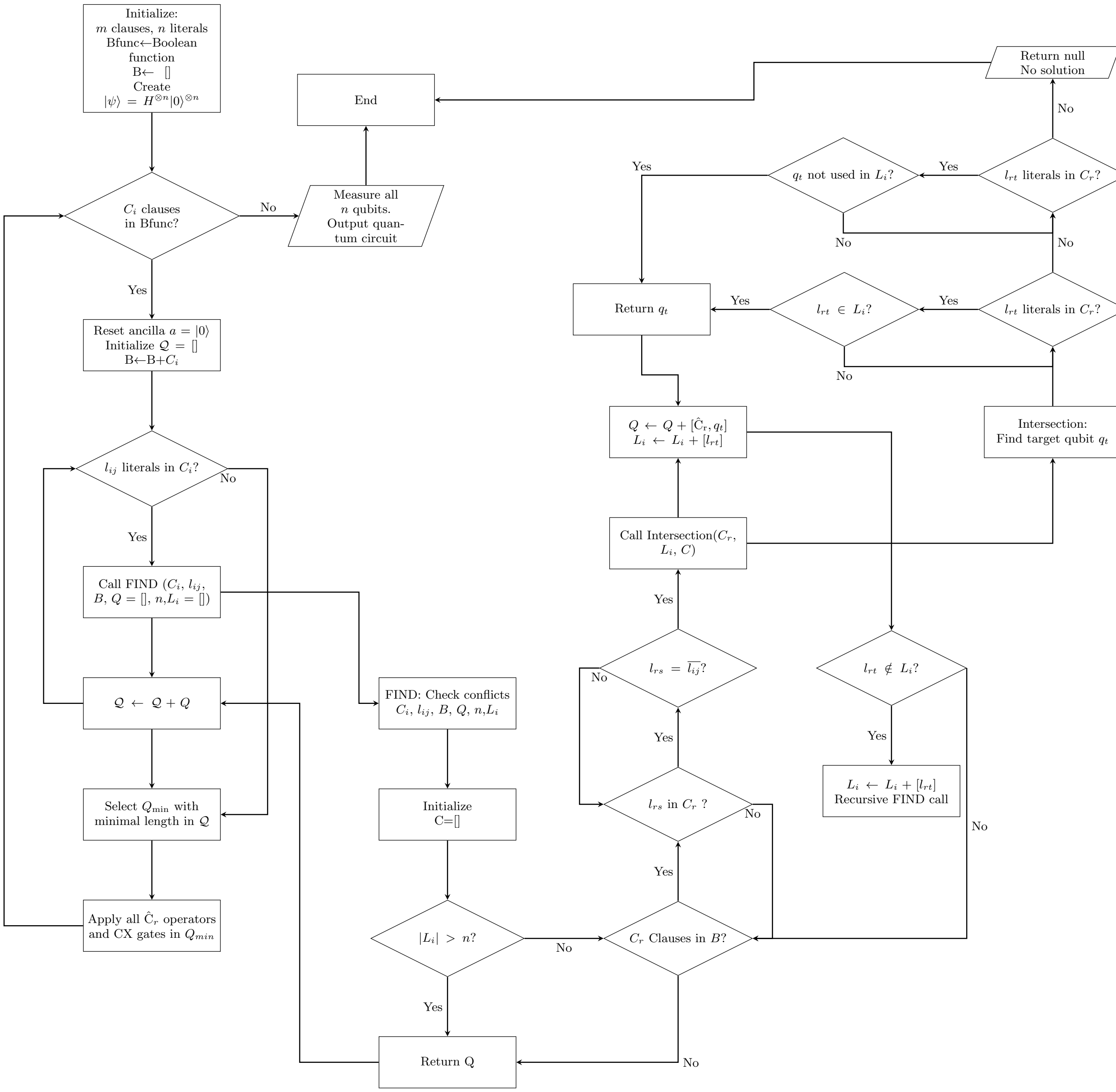

FIG. 5. Flowchart of the complete algorithm

choice of target qubit. We are actively working to provide a more robust implementation of the algorithm. We refer the reader to [9] for the respective GitHub repository to reproduce the work.

III. CONCLUSION

The Boolean satisfiability problem (SAT) is a foundational decision problem in computational complexity and mathematical logic. This problem serves as a cornerstone for understanding NP-completeness and has profound implications across computer science and quantum computing [4]. In this work we have provided a novel quantum solution to the problem. We have used the fact that a single quantum operator can remove all the states that would not satisfy a particular clause from an equal superposition of all states. This allowed us to generate quantum circuits in polynomial time, which would yield a satisfying assignment with the probability of one.

The worst-case classical complexity of generating the quantum circuit is $\mathcal{O}(m^2 \cdot n^3)$. The worst-case quantum circuit depth is $\mathcal{O}(m^2)$. These complexities result in extremely fast run times, especially if we desire a single solution to the problem. The proposed algorithm demonstrates that, by carefully exploiting quantum parallelism and structured state removal, it is possible to solve SAT with polynomial resource scaling, opening new possibilities for practical applications of SAT solvers at industrial scale.

We are working towards a complete proof of the work in a mathematically robust form. We expect some redundancy in the current code framework which would be addressed in the later iterations. Furthermore, we expect machine learning techniques to further enhance the circuit-building process. We expect to reduce the worst-case quantum circuit complexity to $\mathcal{O}(m \cdot n)$.

[1] S. A. Cook, The complexity of theorem-proving procedures, Proceedings of the 3rd Annual ACM Symposium on Theory of Computing , 151 (1971).

[2] M. Davis, G. Logemann, and D. Loveland, A machine program for theorem proving, Communications of the ACM **5**, 394 (1962).

[3] U. Schöning, A probabilistic algorithm for k-sat and constraint satisfaction problems, Proceedings of 40th Annual Symposium on Foundations of Computer Science , 410 (1999).

[4] M. R. Garey and D. S. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness* (W. H. Freeman & Co., USA, 1990).

[5] L. K. Grover, A fast quantum mechanical algorithm for database search, in *Proceedings of the 28th Annual ACM Symposium on Theory of Computing* (1996) pp. 212–219.

[6] A. M. Childs, A. J. Landahl, and P. A. Parrilo, Quantum algorithms for the ordered search problem via semidefinite programming, Physical Review A **75**, 032335 (2007).

[7] S. Boulebnane, A. Montanaro, and A. Lucas, Solving k-sat problems with generalized quantum measurement, arXiv preprint (2024), arXiv:2406.13611 [quant-ph].

[8] Y. Prabhat, S. Thakur, and A. Raina, Structured quantum search algorithm: A quantum leap, arXiv preprint 2504.03426 (2025).

[9] Y. Prabhat, k-sat-problem-solution, `https://github.com/dubeyPraY/k-SAT-problem-solution` (2021), gitHub repository.

[10] A. Javadi-Abhari, M. Treinish, K. Krsulich, C. J. Wood, J. Lishman, J. Gacon, S. Martiel, P. Nation, L. S. Bishop, A. W. Cross, B. R. Johnson, and J. M. Gambetta, Quantum computing with qiskit (2024).