3-SAT Grover Implementation

1 Overview

This implementation addresses 3-SAT by combining quantum counting with Grover search. The input formula is first parsed into a list of clauses and then encoded as a clause–variable matrix, where each row corresponds to a clause and each column corresponds to a variable. This encoding is introduced solely to provide a convenient representation for constructing the quantum oracle.

For a formula with m clauses and n variables, let

$$A = [a_{ij}] \in \{-1, 0, 1\}^{m \times n},$$

where

$$a_{ij} = \begin{cases} 1 & \text{if } x_j \text{ appears as a positive literal in clause } C_i, \\ -1 & \text{if } x_j \text{ appears as a negated literal in clause } C_i, \\ 0 & \text{if } x_j \text{ does not appear in clause } C_i. \end{cases}$$

2 Phase Oracle Construction

The main circuit component is `build_sat_phase_oracle()`. For each clause, the circuit computes whether the clause is satisfied into an ancilla qubit. Operationally, it detects the case in which every literal in the clause is false. Positive literals require temporary X gates so that the multi-controlled gates trigger on the false case. After all clause ancillas are computed, the circuit applies a phase flip conditioned on all clauses being satisfied, then uncomputes the ancillas so they return to $|0\rangle$.

3 Grover Search and Quantum Counting

The program then builds a Grover operator using Qiskit's `grover_operator`, with the reflection applied only to the variable qubits. Because Grover search requires the number of satisfying assignments M, the implementation first estimates M through quantum counting.

Quantum counting applies phase estimation to the Grover operator. It prepares the counting qubits, applies controlled powers of the Grover operator, performs the inverse quantum Fourier transform, measures the counting register, and converts the measured phase into an estimate of M using

$$M = N \sin^2(\theta), \qquad N = 2^n.$$

The rounded estimate is then used to choose the number of Grover iterations before running the final search.

4 Complexity and Resource Requirements

Let n denote the number of Boolean variables, m the number of clauses, $N = 2^n$ the total number of assignments, M the number of satisfying assignments, and t the number of quantum-counting qubits. The SAT oracle uses n variable qubits, m clause ancilla qubits, and one output qubit, so Grover search requires approximately

$$n + m + 1$$

qubits. Quantum counting adds the t-qubit counting register, giving a total of

$$t + n + m + 1$$

qubits.

At a high level, the SAT oracle computes each clause once, applies a multi-controlled phase condition over the clause register, and then uncomputes the clauses. Accordingly, its abstract circuit size is roughly $O(m)$. Although the practical gate count becomes substantially larger after transpilation, because the large multi-controlled gates must be decomposed into elementary operations. One Grover iteration combines this oracle reflection with the diffusion reflection on the n search qubits, so the cost per iteration is approximately

$$O(m + n).$$

The number of Grover iterations is

$$\frac{\pi}{4 \arcsin(\sqrt{M/N})} = O(\sqrt{N/M}),$$

which gives a final Grover-search cost of

$$O\big((m+n)\sqrt{N/M}\big).$$

Quantum counting is more expensive because it applies the controlled powers $G^1, G^2, G^4, \ldots, G^{2^{t-1}}$. Therefore, the total number of Grover-operator applications is

$$1 + 2 + 4 + \cdots + 2^{t-1} = 2^t - 1,$$

and the resulting high-level complexity is approximately

$$O\big((m+n)2^t + t^2\big),$$

where the t^2 term is due to the inverse quantum Fourier transform. Combining the counting stage with the final Grover search yields the overall high-level complexity

$$O\big((m+n)(2^t + \sqrt{N/M}) + t^2\big) = O\big((m+n)(2^t + \sqrt{2^n/M}) + t^2\big).$$

5 Practical Limitations

The main practical limitation is that Aer is a classical simulator, so memory usage and runtime grow rapidly with the total qubit count, namely $t + n + m + 1$, and with the decomposition cost of large multi-controlled gates. In practice, this means that the implementation is best suited to very small example instances with a small number of variables and clauses. Cases with only single digit numbers of variables and clauses are the most realistic. Increasing n enlarges the search space $N = 2^n$, while increasing m both adds clause ancillas and increases the cost of the oracle. Quantum counting is especially expensive because each additional counting qubit doubles the largest controlled Grover power that must be simulated, so even moderate increases in instance size quickly become costly.