SAT Circuits Engine

https://github.com/ohadlev77/sat-circuits-engine

Satisfiability (SAT) problems

- Fundamental in computer science.
- Applicative in many practical and theoretical domains.
- NP-Complete.
- Usually formulated using propositional formulas, e.g:

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

* If there is a combination of boolean values to the variables such that the whole statement is TRUE, we say that the problem is **satisfiable**.

* This particular formula form is called **CNF**.

Satisfiability (SAT) problems

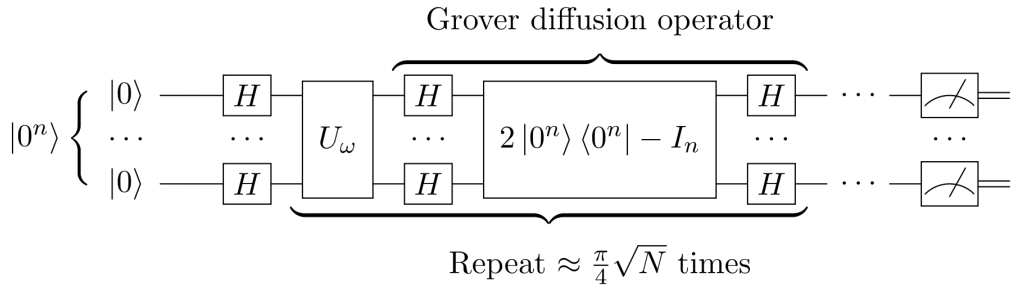
- In particular, a SAT problem formulated using a CNF formula with multiple clauses of length **k** (i.e, with **k** “literals”), is called **k-SAT** problem.
- It has been proven that **k-SAT** problems with **k > 2** are NP-Complete.
- E.g, the following SAT problem is **4-SAT:**

$$(x_1 \vee x_2 \vee x_3 \vee \neg x_4) \wedge (x_1 \vee \neg x_2 \vee x_5 \vee \neg x_6)$$

* Adding enough variables and clauses will eventually turn the SAT problem intractable.

Quantum Speedup

- Classically, solving a general **k-SAT** problem requires exponential time (though many classical heuristics provide significant speedup).
- By encoding the problem onto a quantum circuit it's possible to employ Grover's algorithm and its amplitude amplification generalization to achieve a quadratic speedup:



* Image taken from Wikipedia.

Quantum Circuit Design

- Many consideration factors - width (number of qubits), depth, gate-set, etc.
- Very limited and noisy hardware.

- Optimizations implemented in SAT Circuits Engine (partial list):
  - Decomposing the costly (and typical to Grover's oracle) MCX gate on the fly.
  - Using RCCX gates instead of CCX.
  - Wise ordering of constraints inside Grover's operator.
  - Arithmetics - in Fourier basis, if unavoidable.
  - Reusing the uncomputed auxiliary qubits of Grover's operator to decompose the diffuser's costly MCX gate.

SAT Circuits Engine

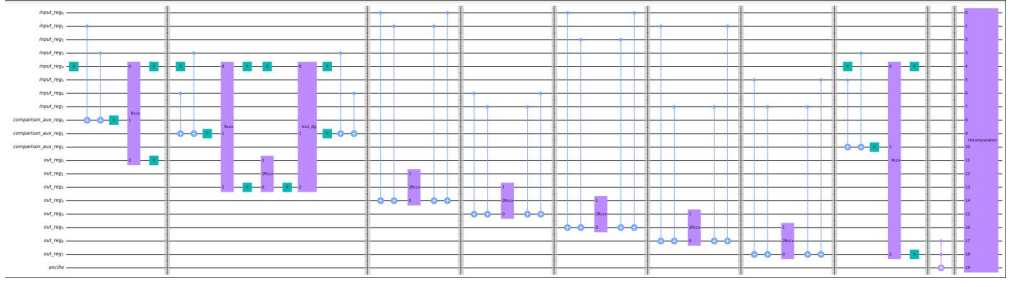
- The main idea - automatic (nearly code-free) and optimized synthesis of quantum circuits for satisfiability problems, of any scale.
- Python-Qiskit based with seamless documentation and exportation of data to universal formats.
- Offers high-level, low-level, CNF and arithmetic input interfaces - the quantum program does more than just SAT-solving.
- Automatically performs circuit-optimizations (above the transpilation level).
- To my knowledge - no other open-source packages offer comparable functionalities and performance.

Input: (x0 != x1) and (x3 != x4) and (x1 != x3) and (x3 != x5) and (x5 != x6) and (x0 != x2) and (x1 != x6) and (x4 != x6)

Output - Solutions

```
High-level format solutions:
Solution 1: x0 = 0, x1 = 1, x2 = 1, x3 = 3, x4 = 1, x5 = 1, x6 = 0
Solution 2: x0 = 0, x1 = 1, x2 = 1, x3 = 2, x4 = 1, x5 = 1, x6 = 0
Solution 3: x0 = 1, x1 = 0, x2 = 0, x3 = 2, x4 = 0, x5 = 0, x6 = 1
Solution 4: x0 = 1, x1 = 0, x2 = 0, x3 = 3, x4 = 0, x5 = 0, x6 = 1
Solution 5: x0 = 0, x1 = 1, x2 = 1, x3 = 0, x4 = 1, x5 = 1, x6 = 0
Solution 6: x0 = 1, x1 = 0, x2 = 0, x3 = 1, x4 = 0, x5 = 0, x6 = 1
```

Output - Grover's Operator



SAT Circuits Engine - Demo

Outlook

- Integrating more dynamical optimization considerations.
- Implementing on real hardware.
- Exploring large problem-instances on classical hardware.
- Dynamic circuit (full) integration.
- Real usefulness.