Satisfiability problem : using Grover’s algorithm

Defining formulas

```
1  c example DIMACS-CNF 3-SAT
2  p cnf 3 5
3  -1 -2 -3 0
4  1 -2 3 0
5  1 2 -3 0
6  1 -2 -3 0
7  -1 2 3 0
```

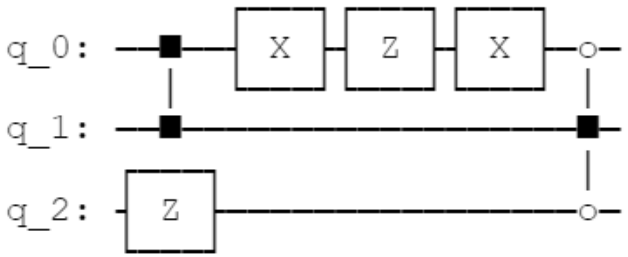
- ¬x1 v ¬x2 v ¬x3
- x1 v ¬x2 v x3
- x1 v x2 v ¬x3
- x1 v ¬x2 v ¬x3
- ¬x1 v x2 v x3

(¬x1 v ¬x2 v ¬x3)^(x1 v ¬x2 v x3)^(x1 v x2 v ¬x3)^
(x1 v ¬x2 v ¬x3)^(¬x1 v x2 v x3)

Oracle

```
oracle = PhaseOracle.from_dimacs_file('3sat.txt')
oracle.draw()
```

```
c example DIMACS-CNF 3-SAT
p cnf 3 5
-1 -2 -3 0
1 -2 3 0
1 2 -3 0
1 -2 -3 0
-1 2 3 0
```



- Phase-flip oracle:
  - flips the phase of the computational basis states corresponding to satisfying assignments of the formula

Verifying clauses

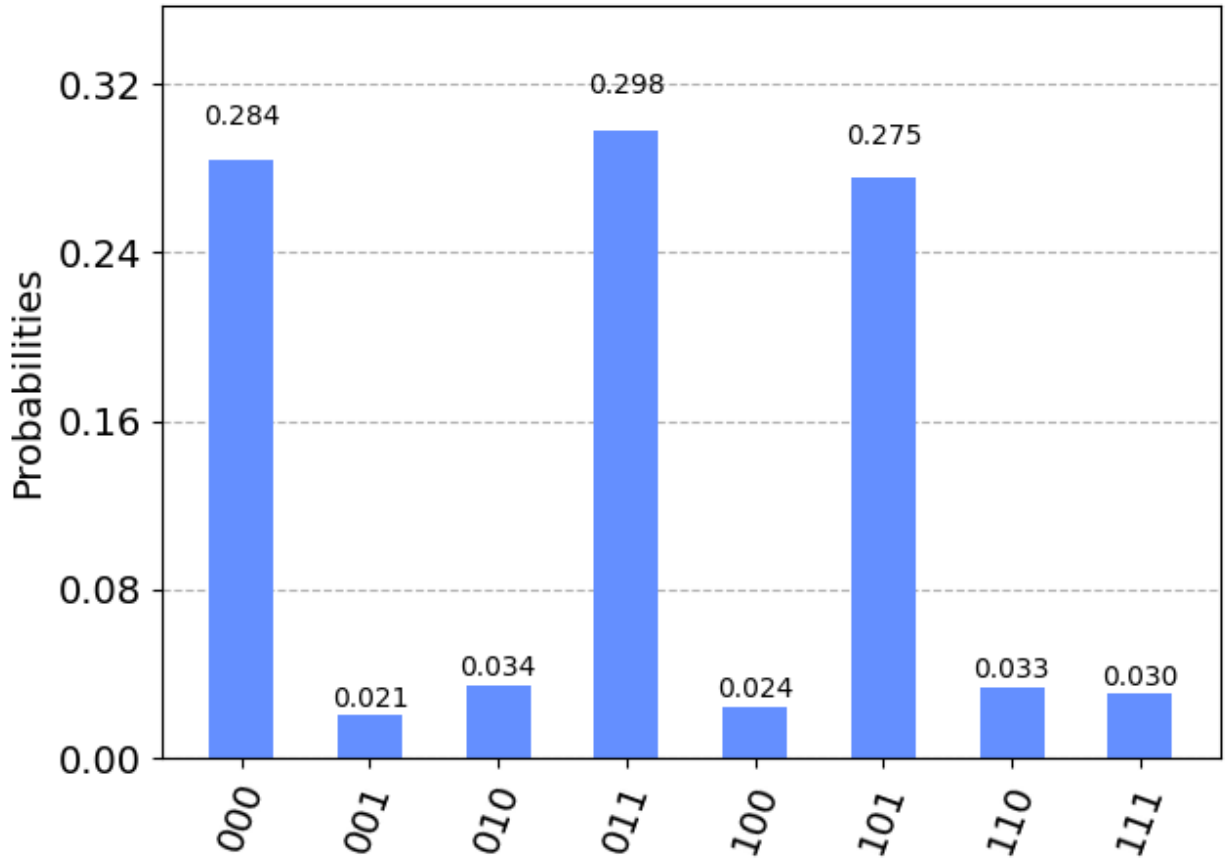
```python
class Verifier():

    def __init__(self, dimacs_file):
        with open(dimacs_file, 'r') as f:
            self.dimacs = f.read()

    def is_correct(self, guess):

        # Convert characters to bools & reverse
        guess = [bool(int(x)) for x in guess][::-1]
        for line in self.dimacs.split('\n'):
            line = line.strip(' 0')
            clause_eval = False
            for literal in line.split(' '):
                if literal in ['p', 'c']:
                    # line is not a clause
                    clause_eval = True
                    break
                if '-' in literal:
                    literal = literal.strip('-')
                    lit_eval = not guess[int(literal)-1]
                else:
                    lit_eval = guess[int(literal)-1]
                clause_eval |= lit_eval
            if clause_eval is False:
                return False
        return True
```

- Init : reads the file
- Is_correct : takes an input and checks whether it is True

A first histogram

```
v = Verifier('3sat.txt')

# Configure backend
backend = Aer.get_backend('aer_simulator')
quantum_instance = QuantumInstance(backend, shots=1024)

# Create a new problem from the phase oracle and the
# verification function
problem = AmplificationProblem(oracle=oracle, is_good_state=v.is_correct)

# Use Grover's algorithm to solve the problem
grover = Grover(quantum_instance=quantum_instance)
result = grover.amplify(problem)
result.top_measurement

plot_histogram(result.circuit_results)
```

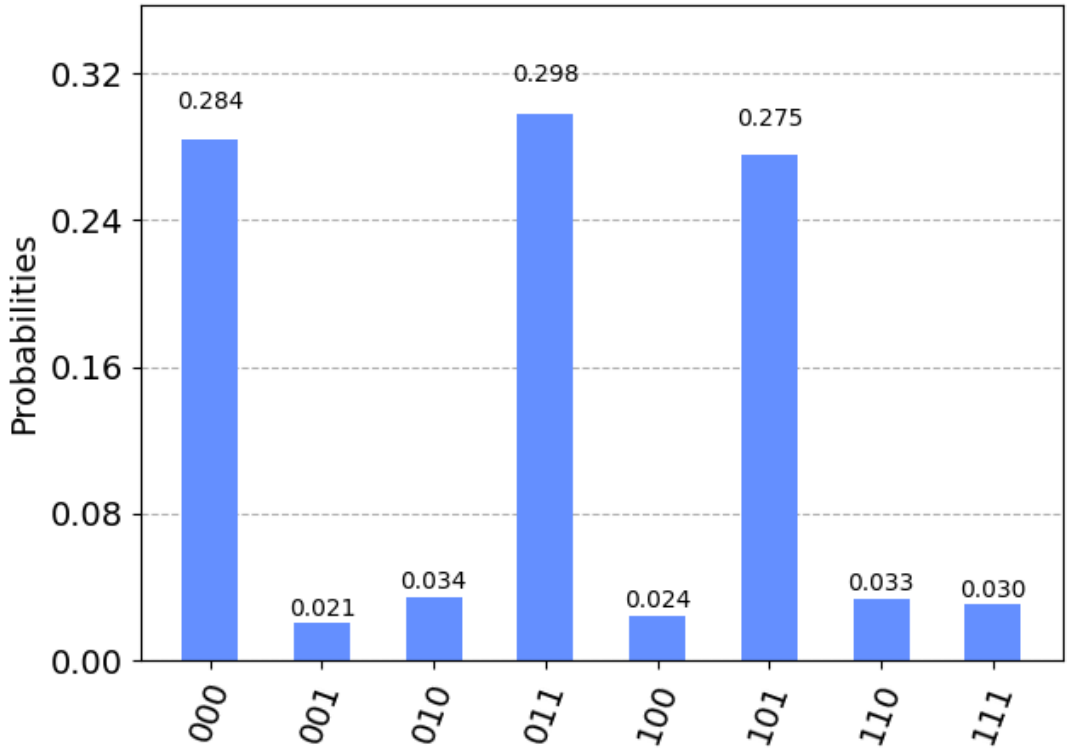
```
1  c example DIMACS-CNF 3-SAT
2  p cnf 3 5
3  -1 -2 -3 0
4  1 -2 3 0
5  1 2 -3 0
6  1 -2 -3 0
7  -1 2 3 0
```



A first histogram

(¬x1 v ¬x2 v ¬x3)^(x1 v ¬x2 v x3)^(x1 v x2 v ¬x3)^ (x1 v ¬x2 v ¬x3)^(¬x1 v x2 v x3)

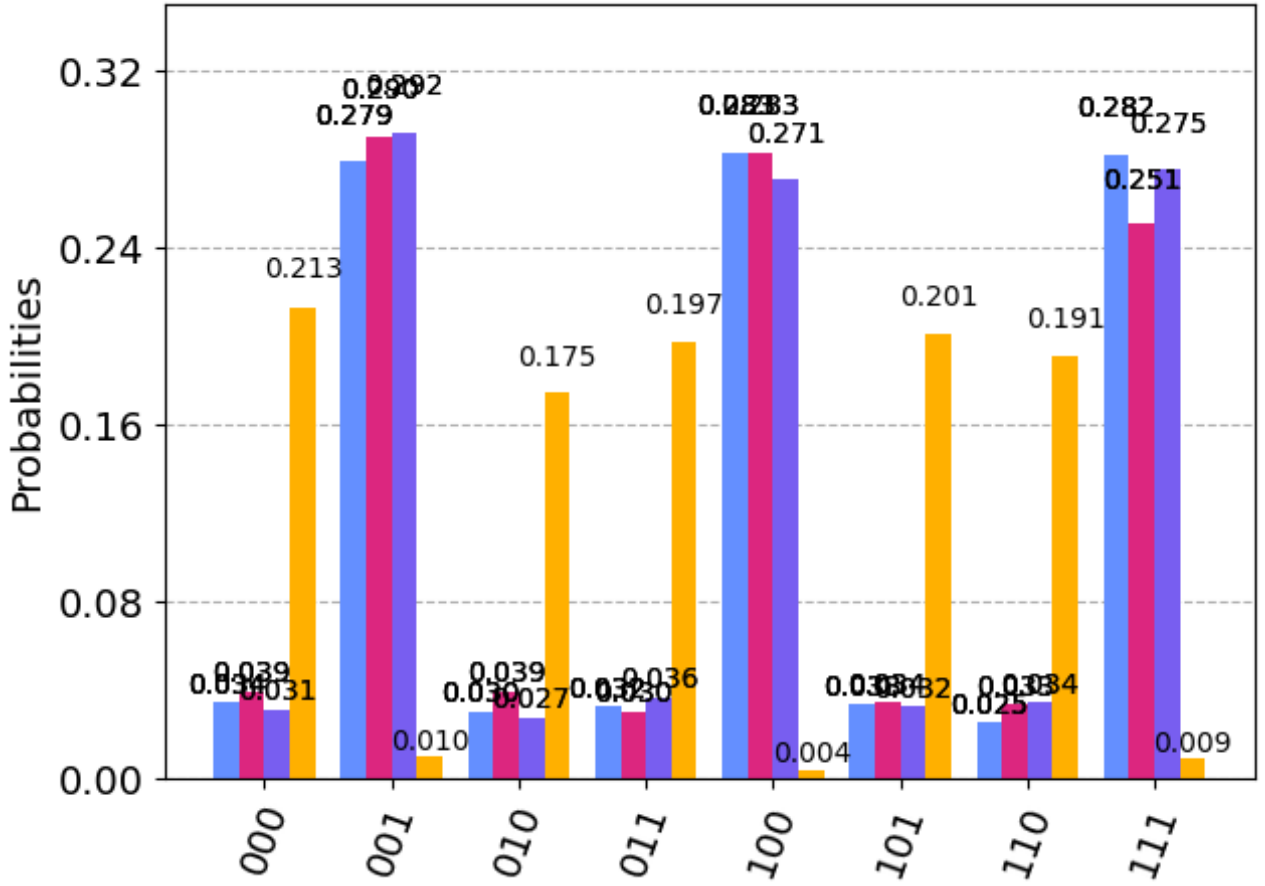
X1	X2	X3	Formula
0	0	0	True
0	0	1	False
0	1	0	False
0	**1**	**1**	**False**
1	0	0	False
1	0	1	True
1	**1**	**0**	**True**
1	1	1	False



Trying out a different formula

(x1 v x2 v ¬x3) ^ (¬x1 v ¬x2 v ¬x3) ^ (¬x1 v x2 v 3)

```
1 p cnf 3 3
2 1 2 -3 0
3 -1 -2 -3 0
4 -1 2 3 0
```

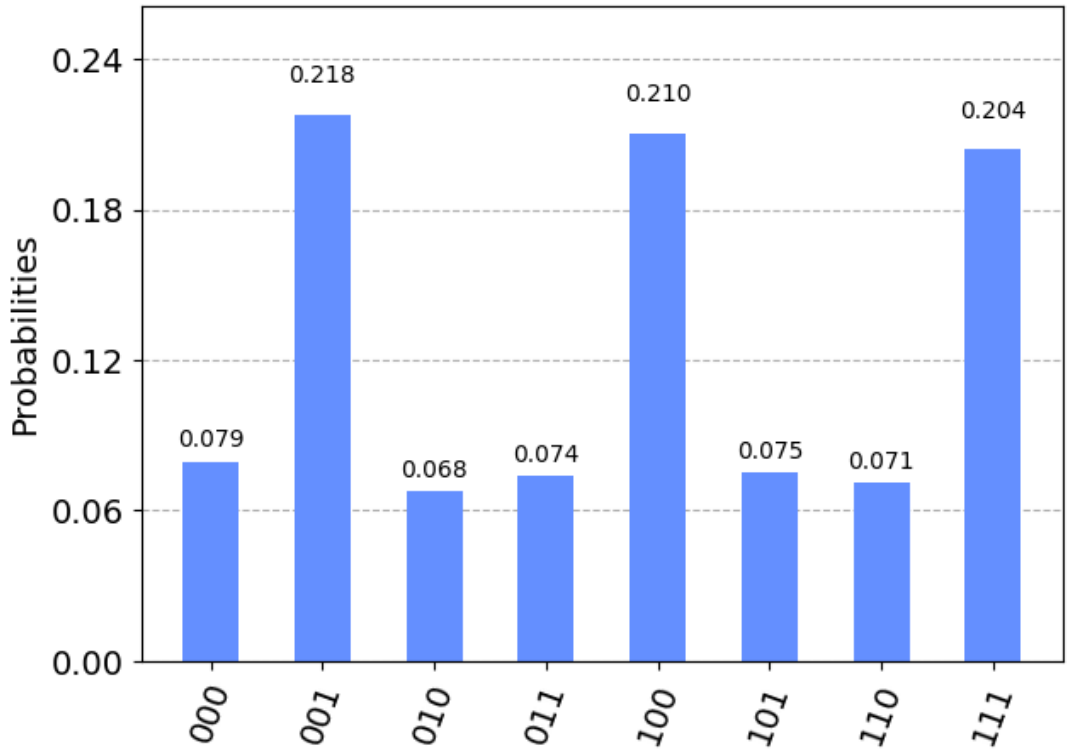


Histogram issues

(x1 v x2 v ¬x3) ^ (¬x1 v ¬x2 v ¬x3) ^ (¬x1 v x2 v 3)

X1	X2	X3	Formula
0	0	0	True
0	0	1	False
0	1	0	True
0	1	1	True
1	0	0	False
1	0	1	True
1	1	0	True
1	1	1	False

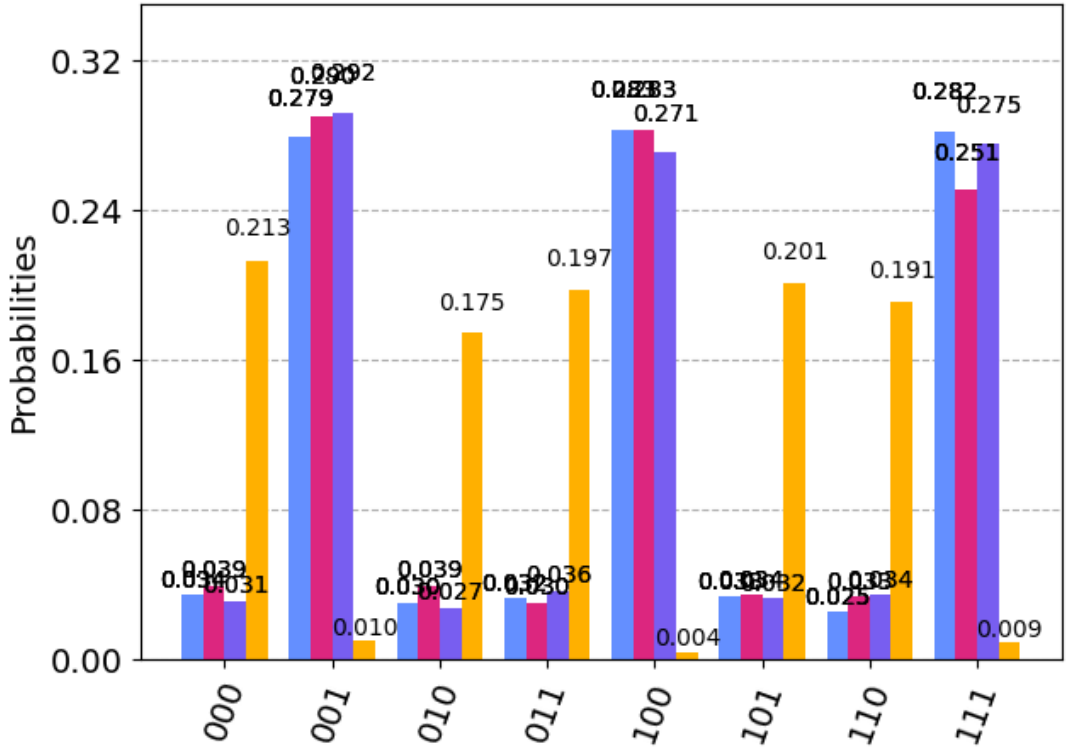
```
d={
    "000": 0,
    "001": 0,
    "010": 0,
    "011": 0,
    "100": 0,
    "101": 0,
    "110": 0,
    "111": 0
}
#print(d)
for i in result.circuit_results:
    for j in i:
        d[j]=d[j]+i[j]
plot_histogram(d)
```



Histograms

(x1 v x2 v ¬x3) ^ (¬x1 v ¬x2 v ¬x3) ^ (¬x1 v x2 v 3)

X1	X2	X3	Formula
0	0	0	True
0	0	1	False
0	1	0	True
0	1	1	True
1	0	0	False
1	0	1	True
1	1	0	True
1	1	1	False

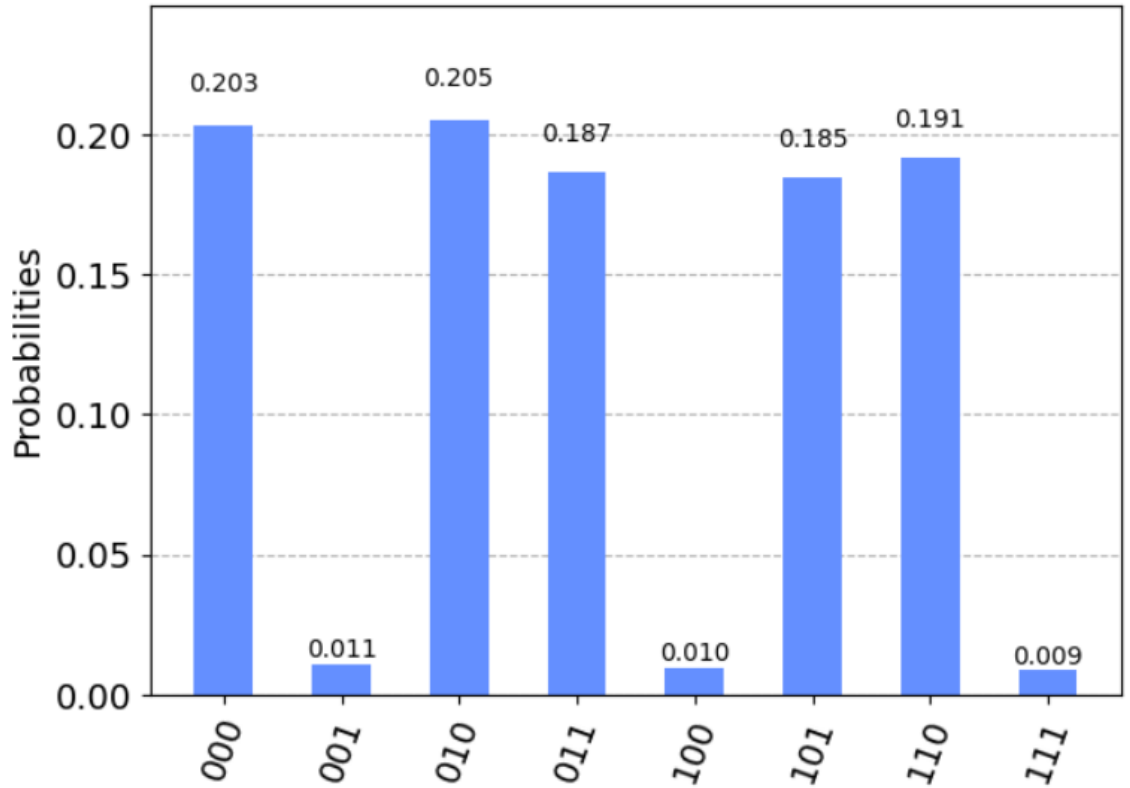


Histograms

(x1 v x2 v ¬x3) ^ (¬x1 v ¬x2 v ¬x3) ^ (¬x1 v x2 v 3)

X1	X2	X3	Formula
0	0	0	True
0	0	1	False
0	1	0	True
0	1	1	True
1	0	0	False
1	0	1	True
1	1	0	True
1	1	1	False

```
plot_histogram(result.circuit_results[3])
```



Another try

```
1 p cnf 3 2
2 -1 -2 3 0
3 1 2 3 0
```

(¬x1 v ¬x2 v x3) ^ (x1 v x2 v x3)