Global Journal of Pure and Applied Mathematics.
ISSN 0973-1768 Volume 20, Number 4 (2024), pp. 695-703
©Research India Publications
https://dx.doi.org/10.37622/GJPAM/20.4.2024.695-703

Quantum Algorithm for 3-SAT Problem by Shor's Fourier Transform with RAM on QCEngine

Toru Fujimura

Art and Physical Education area security office, University of Tsukuba, Ibaraki-branch, Rising Sun Security Service Co., Ltd., 1-1-1, Tennodai, Tsukuba, Ibaraki 305-8577, Japan

Abstract

A quantum algorithm for the 3-SAT problem by the Shor's Fourier transform with the RAM on the QCEngine, and its example are reported. When there are 3 literals with 2 'OR's in each clause, a number of clauses is m, an r-th clause ($1 \leq r \leq m$) is $\mathrm{C}_{u,r}$ (x_1, x_2, x_3, … , x_n) [u is $2^0x_1 + 2^1x_2 + 2^2x_3 + \ldots + 2^{n-1}x_n$. x_1, x_2, …, and x_n are variables.], and $\mathrm{S}(u)_{\max}$ is $\Sigma_{r=1 \to m}\ r \times \mathrm{C}_{u,r}$ (x_1, x_2, x_3, … , x_n), mod($\mathrm{S}(u)_{\max}$) of $\mathrm{S}(u)$ [$\mathrm{S}(u)_{\max}$ is the maximum value of $\mathrm{S}(u)$.] is computed, next, for u, the quantum Fourier transform is done. The complexity of this method is able to be several times.

Keywords: Quantum algorithm, 3-SAT problem, Shor's Fourier transform, RAM, QCEngine.

AMS subject classification: Primary 81-08; Secondary 81-10, 68Q12.

1. Introduction

Cook discussed the complexity of the 3-SAT problem. [1] Quantum computer's example of the 3-SAT problem is reported by Johnston, Harrigan, and Gimeno-Segovia with QCEngine (free on-line quantum computer simulator). [2] Fujimura discussed a quantum algorithm for the 3-SAT problem by the Grover iteration with the CCCNOT gate (= control Toffoli gate) on the QCEngine. [3]

According to my advanced study, when the Shor’s Fourier transform for the 3-SAT problem is used, the complexity of the 3-SAT problem is able to be several times.

Therefore, because the quantum algorithm for the 3-SAT problem is examined by the Shor’s Fourier transform with the RAM on the QCEngine, its result is reported.

2. 3-SAT Problem

In the 3-SAT problem, it is assumed that (i) each value of n variables becomes “TRUE” or “FALSE”, “~” is “NOT”, “V” is “OR”, “&” is “AND”, (ii) “V”, “~”, and 3 different variables are included in each parentheses (= clause) that are connected by “&”. If a value of logical formula by the literals and the logical connectives is “TRUE”, it is decided whether there is at least one combination of values of the variables or not. [1-3]

3. Quantum Algorithm

The following conditions are assumed. (I) Each value of variables x_1, x_2, x_3, … , and x_n becomes “TRUE” [= 1], or “FALSE” [= 0]. “~” is “NOT”. “V” is “OR”. “&” is “AND”. For example, it is assumed in this algorithm that (1 V 1 V 1), (1 V 1 V 0), and (1 V 0 V 0) become 1, and (0 V 0 V 0) becomes 0. (II) “V”, “~”, and 3 different variables in x_1, x_2, x_3, … , and x_n are included in each clause, and then the clauses are connected by “&”. In these conditions, if a value of logical formula by the literals, and the operators is “TRUE”, it is searched whether there is at least one combination of values of the variables or not. It is assumed that n is number of qubits, u is $2^0x_1 + 2^1x_2 + 2^2x_3 + \dots + 2^{n-1}x_n$, a number of clauses is m, an r-th clause ($1 \le r \le m$) is $\mathrm{C}_{u,r}$ (x_1, x_2, x_3, … , x_n), $\mathrm{S}(u)$ is $\Sigma_{\,r=1\to m}\, r\times\mathrm{C}_{u,r}$ (x_1, x_2, x_3, … , x_n), and $\mathrm{S}(u)_{\max}$ is (the maximum value of $\mathrm{S}(u)$) $= (m + 1)m/2 = k$.

First of all, query quantum registers $|x_i\rangle$[$1 \le i \le n$. i is an integer. n is the number of variables in logical formula.], work1 quantum registers $|w_{1,j}\rangle$ [$1 \le j \le t$. j, and t are integers. t is a necessary number for $\mathrm{S}(u)_{\max} \le 2^t$.], work2 quantum registers $|w_{2,p}\rangle$ [1

$\leq p \leq t + 1$. p is an integer. +1 is a qubit for the negative integer. [2]], and ancilla quantum qubit $|a\rangle$ are prepared.

Step 1: The r data are introduced to the RAM [2].

Step 2: Each qubit of $|x_i\rangle$, $|w_{1,j}\rangle$, $|w_{2,p}\rangle$, and $|a\rangle$ is set $|0\rangle$.

Step 3: The Hadamard gate $\boxed{\mathrm{H}}$ [2-8] acts on each qubit of $|x_i\rangle$. It changes them for entangled states.

Step 4: Each clause is presented by $|x_i\rangle$, $|w_{2,p}\rangle$, add gate, and quantum operators. For $|x_i\rangle$, RAM[r - 1] [RAM has r data of $0 \rightarrow (\mathrm{m} - 1)$.] is incremented in $|w_{2,p}\rangle$. In a function, $\mathrm{S}(u) = \Sigma_{r=1\rightarrow m}\, r\times\mathrm{C}_{u,r}\,(x_1, x_2, x_3, \ldots, x_n)$ is computed. This operation makes entangled data base.

Step 5: For $|w_{2,p}\rangle$, mod(k) [$k = \mathrm{S}(u)_{\max} = (m + 1)m/2$] is done, where mod($k$) is made by subtraction and addition in this program. [2] Therefore, the subtraction and the addition are done, where work1 quantum registers are added work2 quantum registers, and the uncompute is done.

Step 6: For $|x_i\rangle$, the quantum Fourier transform (= QFT) [2, 4, 5, 8] is done.

Step 7: For $|x_i\rangle$, and $|w_{1,j}\rangle$, the proves are done.

Step 8: For $|x_i\rangle$, the read is done.

Step 9: A number of spikes is estimated by the function (https: //oreilly-qc. github. io? p = 12-4 [2]), where the function estimate_num_spikes (spike, range) [spike: read value, range: 2^n] is used.

Step 10: From candidates of the number of spikes, the repeat period P is obtained.

Step 11: From $\mathrm{P} = 2^0x_1 + 2^1x_2 + 2^2x_3 + \ldots + 2^{n-1}x_n$, when there is S(P) is $\Sigma_{r=1\rightarrow m}\, r\times\mathrm{C}_{\mathrm{P},r}\,(x_1, x_2, x_3, \ldots, x_n) = \mathrm{S}(u)_{\max} = k$, it is answer [number of combination of (value of logical formula) = 1].

4. Example of Numerical Computation

For example at $n = 5$, it is assumed that logical formula : $(x_3 \vee x_4 \vee x_5)$ & $(\sim x_1 \vee x_2 \vee$

x_3) & ($\sim x_3$ V x_4 V x_5) & (x_3 V $\sim x_4$ V x_5) & ($\sim x_2$ V x_3 V $\sim x_5$) & ($\sim x_3$ V $\sim x_4$ V x_5) & ($\sim x_3$ V x_4 V $\sim x_5$) & (x_3 V $\sim x_4$ V $\sim x_5$) & ($\sim x_3$ V $\sim x_4$ V $\sim x_5$), each value of $x_{1\sim5}$: $x_1 = x_2 = x_3 = x_4 = 0$, $x_5 = 1$, $m = 9$, $t = 6$, and $k = (m + 1)m/2 = 45$.

An example of program on the QCEngine is the following.

```
10 var a = [1, 2, 3, 4, 5, 6, 7, 8, 9]; // RAM_a

20 var query_qubits = 5;

30 var work1_qubits = 6;

40 var work2_qubits = 7;

50 var ancilla_qubit = 1;

60 qc.reset(query_qubits + work1_qubits + work2_qubits + ancilla_qubit);

70 var query = qint.new(query_qubits, 'query');

80 var work1 = qint.new(work1_qubits, 'work1');

90 var work2 = qint.new(work2_qubits, 'work2');

100 var ancilla = qint.new(ancilla_qubit, 'ancilla');

110 qc.label('q'); // set query

120 query.write(0);

130 query.hadamard();

140 qc.label(' ');

150 qc.label('w1'); // set work1

160 work1.write(0);

170 qc.label('w2'); // set work2

180 work2.write(0);

190 qc.label('a'); // set ancilla

200 ancilla.write(0);

210 qc.print(' RAM before increment : ' + a + '¥n');

220 var query16 = 16;

230 var k = 45;
```

```
240 var work1_0 = 0;
250 qc.label('increment');
260 qc.not(query.bits(0x4)|query.bits(0x8)|query.bits(0x10));
270 work2.add(a[0],query.bits(0x4)|query.bits(0x8)|query.bits(0x10));
280 qc.not(query.bits(0x4)|query.bits(0x8)|query.bits(0x10));
290 qc.not(query.bits(0x2)|query.bits(0x4));
300 work2.add(a[1],query.bits(0x1)|query.bits(0x2)|query.bits(0x4));
310 qc.not(query.bits(0x2)|query.bits(0x4));
320 qc.not(query.bits(0x8)|query.bits(0x10));
330 work2.add(a[2],query.bits(0x4)|query.bits(0x8)|query.bits(0x10));
340 qc.not(query.bits(0x8)|query.bits(0x10));
350 qc.not(query.bits(0x4)|query.bits(0x10));
360 work2.add(a[3],query.bits(0x4)|query.bits(0x8)|query.bits(0x10));
370 qc.not(query.bits(0x4)|query.bits(0x10));
380 qc.not(query.bits(0x4));
390 work2.add(a[4],query.bits(0x2)|query.bits(0x4)|query.bits(0x10));
400 qc.not(query.bits(0x4));
410 qc.not(query.bits(0x10));
420 work2.add(a[5],query.bits(0x4)|query.bits(0x8)|query.bits(0x10));
430 qc.not(query.bits(0x10));
440 qc.not(query.bits(0x8));
450 work2.add(a[6],query.bits(0x4)|query.bits(0x8)|query.bits(0x10));
460 qc.not(query.bits(0x8));
470 qc.not(query.bits(0x4));
480 work2.add(a[7],query.bits(0x4)|query.bits(0x8)|query.bits(0x10));
490 qc.not(query.bits(0x4));
```

```
work2.add(a[8],query.bits(0x4)|query.bits(0x8)|query.bits(0x10));
qc.label('mod(' + k + ')');
work2.subtract(k);
qc.cnot(ancilla.bits(0x1), work2.bits(0x40));
work2.add(k,ancilla.bits(0x1));
work1.add(work2);
qc.label('uncompute');
work2.subtract(k,ancilla.bits(0x1));
qc.cnot(ancilla.bits(0x1),work2.bits(0x40));
work2.add(k);
work2.subtract(a[8],query.bits(0x4)|query.bits(0x8)|query.bits(0x10));
qc.not(query.bits(0x4));
work2.subtract(a[7],query.bits(0x4)|query.bits(0x8)|query.bits(0x10));
qc.not(query.bits(0x4));
qc.not(query.bits(0x8));
work2.subtract(a[6],query.bits(0x4)|query.bits(0x8)|query.bits(0x10));
qc.not(query.bits(0x8));
qc.not(query.bits(0x10));
work2.subtract(a[5],query.bits(0x4)|query.bits(0x8)|query.bits(0x10));
qc.not(query.bits(0x10));
qc.not(query.bits(0x4));
work2.subtract(a[4],query.bits(0x2)|query.bits(0x4)|query.bits(0x10));
qc.not(query.bits(0x4));
qc.not(query.bits(0x4)|query.bits(0x10));
work2.subtract(a[3],query.bits(0x4)|query.bits(0x8)|query.bits(0x10));
qc.not(query.bits(0x4)|query.bits(0x10));
```

```
760 qc.not(query.bits(0x8)|query.bits(0x10));
770 work2.subtract(a[2],query.bits(0x4)|query.bits(0x8)|query.bits(0x10));
780 qc.not(query.bits(0x8)|query.bits(0x10));
790 qc.not(query.bits(0x2)|query.bits(0x4));
800 work2.subtract(a[1],query.bits(0x1)|query.bits(0x2)|query.bits(0x4));
810 qc.not(query.bits(0x2)|query.bits(0x4));
820 qc.not(query.bits(0x4)|query.bits(0x8)|query.bits(0x10));
830 work2.subtract(a[0],query.bits(0x4)|query.bits(0x8)|query.bits(0x10));
840 qc.not(query.bits(0x4)|query.bits(0x8)|query.bits(0x10));
850 qc.label('QFT');
860 query.QFT();
870 var prob16 = 0;
880 prob16 += query.peekProbability(query16);
890 // Print output query-Prob
900 qc.print(' Prob_query16: ' + prob16);
910 var prob0 = 0;
920 prob0 += work1.peekProbability(work1_0);
930 // Print output work1-Prob
940 qc.print(' Prob_work1_0: ' + prob0);
950 //read
960 qc.label('Rq');
970 var b2 = query.read();
980 // Print output result
990 qc.print(' Read query = ' + b2 +'.');
1000 // end
```

When this program is copied on Programming Quantum Computers https: //oreilly-qc.

github. io/# [free on-line quantum computation simulator QCEngine] [2], you can run it. [Caution!: Please delate the line numbers.]

A result of this program is the following.

The probability probe value of $|w_{1,j}\rangle = 0$: 0.031250 (= 1/32).

The probability probe value of $|x_i\rangle = 16$: $\approx$ 0.0078125.

The example of 10 times test : The read value of $|x_i\rangle$; R_q = 22, 1, 21, 30, 0, 21, 28, 25, 3, 0. (=spike)

The candidates of number of spikes are estimated by the function [the function estimate_num_spikes (spike, range) [spike : read value, range : $2^n = 2^5 = 32$]] : $R_q \rightarrow$ candidates ; 22 $\rightarrow$ 3, 6, 10, 13, 16 ; 1 $\rightarrow$ nothingness ; 21 $\rightarrow$ 3, 6, 9, 12, 15, 17 ; 30 $\rightarrow$ 16 ; 0 $\rightarrow$ nothingness ; 21 $\rightarrow$ 3, 6, 9, 12, 15, 17 ; 28 $\rightarrow$ 8, 16, 24 ; 25 $\rightarrow$ 5, 9, 18, 23 ; 3 $\rightarrow$ 11, 21 ; 0 $\rightarrow$ nothingness.

When u is 16 ($2^0x_1 + 2^1x_2 + 2^2x_3 + 2^3x_4 + 2^4x_5 = 2^0\times0 + 2^1\times0 + 2^2\times0 + 2^3\times0 + 2^4\times1 = 16$), the value of logical formula is 1. Therefore, it is the answer.

5. Discussion

In the 3-SAT problem, when [(the logical formula) = 1] is obtained, there is only one combination.

Therefore, the search is difficult.

In the section 4, n is 5. And then, in n variables, [(the logical formula) = 1] of combination of variables is selected. When N is 2^n, in the Grover's method, the complexity is $N^{1/2} = 2^{5/2} \approx 6$, in the Shor's Fourier transform, it is $10/3 \approx 3$.

In this range, the Shor's Fourier transform is less than the complexity of the Grover's method.

6. Summary

The quantum algorithm for the 3-SAT problem by the Shor's Fourier transform with the RAM on the QCEngine, and its example are reported.

The complexity of this method is several times.

I will apply this method for other problems.

References

[1] Cook, S. A., 1971, "The complexity of theorem proving procedures," Proc. 3rd Annu. ACM Symp. Theory of Computing, pp.151-158.

[2] Johnston, E.R., Harrigan, N., and Gimeno-Segovia, M., 2019, Programming Quantum Computers, O'Reilly, ISBN 978-1-492-03968-6.

[3] Fujimura, T., 2023, "Quantum algorithm for 3-SAT problem by Grover iteration with CCCNOT gate (= control Toffoli gate) on QCEngine," Glob. J. Pure Appl. Math., **19**, 151-156.

[4] Takeuchi, S., 2005, Ryoshi Konpyuta (Quantum Computer), Kodansha, Tokyo, Japan [in Japanese].

[5] Shor, P. W., 1994, "Algorithms for quantum computation : discrete logarithms and factoring," Proc. 35th Annu. Symp. Foundations of Computer Science, IEEE, pp.124-134.

[6] Grover, L. K., 1996, "A fast quantum mechanical algorithm for database search," Proc. 28th Annu. ACM Symp. Theory of Computing, pp.212-219.

[7] Grover, L. K., 1998, "A framework for fast quantum mechanical algorithms," Proc. 30th Annu. ACM Symp. Theory of Computing, pp.53-62.

[8] Miyano, K., and Furusawa, A., 2008, Ryoshi Konpyuta Nyumon (An Introduction to Quantum Computation), Nipponhyoronsha, Tokyo, Japan [in Japanese].