

Quantum Algorithm for 3-SAT Problem by Quantum Fourier Transform with CCCNOT Gate (= Control Toffoli Gate) 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
E-mail: tfujimura8@gmail.com*

Abstract

A quantum algorithm for the 3-SAT problem by the quantum Fourier transform (= QFT) with the CCCNOT gate (= control Toffoli gate) on the QCEngine, and its example are reported. In this method, there are 3 literals with 2 'OR's in each clause that is used the CCCNOT gate. The complexity of this QFT method is able to be several times, because this method contains the calculated data set of marked term. This method is simple and high speed.

Keywords: Quantum algorithm, 3-SAT problem, quantum Fourier transform (= QFT), CCCNOT gate (= control Toffoli gate), QCEngine.

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

1. Introduction

The complexity of the 3-SAT problem has been discussed by Cook. [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 quantum Fourier transform (= QFT) with the CCCNOT gate on the QCEngine is used, the complexity of the 3-SAT problem is able to be several times, because this method contains the calculated data set of marked term (= M). This method is simple and high speed.

Therefore, because the quantum algorithm for the 3-SAT problem is examined by the QFT with the CCCNOT gate 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”, “ \sim ” is “NOT”, “V” is “OR”, “&” is “AND”, (ii) “V”, “ \sim ”, 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, \dots, x_{n-1} , and x_n becomes “TRUE” [= 1] or “FALSE” [= 0]. “ \sim ” 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”, “ \sim ”, and 3 different variables in x_1, x_2, \dots, x_{n-1} , 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. M is $2^0 m_1 + 2^1 m_2 + 2^2 m_3 + \dots + 2^{n-1} m_n$, and $m_1, m_2, m_3, \dots, m_n$ are integers.

First of all, query quantum registers $|x_i\rangle$, work1 quantum registers $|y_i\rangle$, [$3 \leq i \leq n$. i is an integer. n is the number of variables in logical formula.], work2 quantum registers $|z_j\rangle$ [$1 \leq j \leq u$. j and u are integers. u is a necessary number for work.], and ancilla quantum register $|a\rangle$ are prepared.

Step 0: Please confirm whether 0 or 1 is answer or not [for example, (x_1, x_2, \dots, x_n)

$= (0, 0, \dots, 0)$ or $(1, 0, \dots, 0)$]. When it isn't answer, the following will be done.

Step 1: Each qubit of $|x_i\rangle |y_i\rangle |z_j\rangle$ and $|a\rangle$ is set $|0\rangle$.

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

Step 3: The value of query quantum registers is added for work1 quantum registers.

Step 4: Each clause is presented by $|x_i\rangle, |z_j\rangle$, CCCNOT gate, and quantum operators.

Step 5: When all of $|z_j\rangle$ are 1, $|a\rangle$ is operated "NOT".

Step 6: When $|a\rangle$ is 1, the value of query quantum registers is subtracted for work1 quantum registers

Step 7: The uncompute is done for step 4 and step 5.

Step 8: The QFT is done for the query quantum registers.

Step 9: For the query quantum registers and the work1 quantum registers, the probability probes are done.

Step 10: The query quantum registers is read.

Step 11: The R_q (= read value) and the range (= 2^n) are used for the calculated data set of M. This data set is made by $2^n/M = S$, $1 \leq [kS] \leq 2^n-1$, $k =$ (natural number), and $[kS] =$ (kS is rounded to one decimal place.).

Step 12: From the calculated data set of M, candidates of M is obtained.

Step 13: From $M = 2^0m_1 + 2^1m_2 + 2^2m_3 + \dots + 2^{n-1}m_n$, when $(x_1, x_2, x_3, \dots, x_n)$ is $(m_1, m_2, m_3, \dots, m_n)$, and the value of logical formula is 1, the answer is $(x_1, x_2, x_3, \dots, x_n) = (m_1, m_2, m_3, \dots, m_n)$, where there is the answer $A = 2^0x_1 + 2^1x_2 + 2^2x_3 + \dots + 2^{n-1}x_n$.

The one-marked-term is obtained.

4. Example of Numerical Computation

For example at $n = 5$, it is assumed that the $M = 16$ (= A), logical formula : $(x_3 \vee x_4 \vee x_5) \& (\sim x_1 \vee x_2 \vee x_3) \& (\sim x_3 \vee x_4 \vee x_5) \& (x_3 \vee \sim x_4 \vee x_5) \& (\sim x_2 \vee x_3 \vee \sim x_5) \& (\sim x_3 \vee \sim x_4 \vee x_5) \& (\sim x_3 \vee x_4 \vee \sim x_5) \& (x_3 \vee \sim x_4 \vee \sim x_5) \& (\sim x_3 \vee \sim x_4 \vee \sim x_5)$, each value of x_{1-5} : $x_1 = x_2 = x_3 = x_4 = 0$, $x_5 = 1$, work2 qubits = $u = 9$, and ancilla qubit.

An example of program on the QCEngine is the following.

```
10 var query_qubits = 5;
20 var work1_qubits = 5;
30 var work2_qubits = 9;
40 var ancilla_qubit = 1;
50 qc.reset(query_qubits + work1_qubits + work2_qubits + ancilla_qubit);
60 var query = qint.new(query_qubits, 'query');
70 var work1 = qint.new(work1_qubits, 'work1');
80 var work2 = qint.new(work2_qubits, 'work2');
90 var ancilla = qint.new(ancilla_qubit, 'ancilla');
100 qc.label('q'); // set query
110 query.write(0);
120 query.hadamard();
130 qc.label(' ');
140 qc.label('w1'); // set work1
150 work1.write(0);
160 qc.label('w2'); // set work2
170 work2.write(0);
180 qc.label('a'); // set ancilla
190 ancilla.write(0);
200 var query16 = 16;
210 var work1_0 = 0;
220 qc.label(('w1 += q'); // work1 += query
230 work1.add(query);
240 qc.label('Gate');
250 qc.not(query.bits(0x4)|query.bits(0x8)|query.bits(0x10));
260 qc.cnot(work2.bits(0x1),query.bits(0x4)|query.bits(0x8)|query.bits(0x10));
```

```
270 qc.not(query.bits(0x4)|query.bits(0x8)|query.bits(0x10)|work2.bits(0x1));
280 qc.not(query.bits(0x2)|query.bits(0x4));
290 qc.cnot(wor2k.bits(0x2),query.bits(0x1)|query.bits(0x2)|query.bits(0x4));
300 qc.not(query.bits(0x2)|query.bits(0x4)|work2.bits(0x2));
310 qc.not(query.bits(0x8)|query.bits(0x10));
320 qc.cnot(work2.bits(0x4),query.bits(0x4)|query.bits(0x8)|query.bits(0x10));
330 qc.not(query.bits(0x8)|query.bits(0x10)|work2.bits(0x4));
340 qc.not(query.bits(0x4)|query.bits(0x10));
350 qc.cnot(work2.bits(0x8),query.bits(0x4)|query.bits(0x8)|query.bits(0x10));
360 qc.not(query.bits(0x4)|query.bits(0x10)|work2.bits(0x8));
370 qc.not(query.bits(0x4));
380 qc.cnot(work2.bits(0x10),query.bits(0x2)|query.bits(0x4)|query.bits(0x10));
390 qc.not(query.bits(0x4) |work2.bits(0x10));
400 qc.not(query.bits(0x10));
410 qc.cnot(work2.bits(0x20),query.bits(0x4)|query.bits(0x8)|query.bits(0x10));
420 qc.not(query.bits(0x10)|work2.bits(0x20));
430 qc.not(query.bits(0x8));
440 qc.cnot(work2.bits(0x40),query.bits(0x4)|query.bits(0x8)|query.bits(0x10));
450 qc.not(query.bits(0x8)|work2.bits(0x40));
460 qc.not(query.bits(0x4));
470 qc.cnot(work2.bits(0x80),query.bits(0x4)|query.bits(0x8)|query.bits(0x10));
480 qc.not(query.bits(0x4)|work2.bits(0x80));
490 qc.cnot(work2.bits(0x100),query.bits(0x4)|query.bits(0x8)|query.bits(0x10));
500 qc.not(work2.bits(0x100));
510 qc.label('a CN w2'); // ancilla CNOT work2
520 qc.cnot(ancilla.bits(0x1),work2.bits());
```

```
530 qc.label('w1 -= q'); // work1 -= query at ancilla = 1
540 work1.subtract (query,ancilla.bits(0x1));
550 qc.label('uncompute');
560 qc.cnot(ancilla.bits(0x1),work2.bits());
570 qc.not(work2.bits(0x100));
580 qc.cnot(work2.bits(0x100),query.bits(0x4)|query.bits(0x8)|query.bits(0x10));
590 qc.not(query.bits(0x4)|work2.bits(0x80));
600 qc.cnot(work2.bits(0x80),query.bits(0x4)|query.bits(0x8)|query.bits(0x10));
610 qc.not(query.bits(0x4));
620 qc.not(query.bits(0x8)|work2.bits(0x40));
630 qc.cnot(work2.bits(0x40),query.bits(0x4)|query.bits(0x8)|query.bits(0x10));
640 qc.not(query.bits(0x8));
650 qc.not(query.bits(0x10)|work2.bits(0x20));
660 qc.cnot(work2.bits(0x20),query.bits(0x4)|query.bits(0x8)|query.bits(0x10));
670 qc.not(query.bits(0x10));
680 qc.not(query.bits(0x4)|work2.bits(0x10));
690 qc.cnot(work2.bits(0x10),query.bits(0x2)|query.bits(0x4)|query.bits(0x10));
700 qc.not(query.bits(0x4));
710 qc.not(query.bits(0x4)|query.bits(0x10)|work2.bits(0x8));
720 qc.cnot(work2.bits(0x8),query.bits(0x4)|query.bits(0x8)|query.bits(0x10));
730 qc.not(query.bits(0x4)|query.bits(0x10));
740 qc.not(query.bits(0x8)|query.bits(0x10)|work2.bits(0x4));
750 qc.cnot(work2.bits(0x4),query.bits(0x4)|query.bits(0x8)|query.bits(0x10));
760 qc.not(query.bits(0x8)|query.bits(0x10));
770 qc.not(query.bits(0x2)|query.bits(0x4)|work2.bits(0x2));
780 qc.cnot(work2.bits(0x2),query.bits(0x1)|query.bits(0x2)|query.bits(0x4));
```

```
790 qc.not(query.bits(0x2)|query.bits(0x4));
800 qc.not(query.bits(0x4)|query.bits(0x8)|query.bits(0x10)|work2.bits(0x1));
810 qc.cnot(work2.bits(0x1),query.bits(0x4)|query.bits(0x8)|query.bits(0x10));
820 qc.not(query.bits(0x4)|query.bits(0x8)|query.bits(0x10));
830 qc.label('QFT');
840 query.QFT();
850 var prob0 = 0;
860 prob0 += work1.peekProbability(work1_0); // work1_0 = 0
870 // Print output work1-Prob
880 qc.print(' Prob_work1_0: ' + prob0);
890 var prob16 = 0;
900 prob16 += query.peekProbability(query16); // query16 = 0
910 // Print output query-Prob
920 qc.print(' Prob_query16: ' + prob16);
930 //read
940 qc.label('Rq');
950 var b2 = query.read();
960 // Print output result
970 qc.print(' Read query = ' + b2 + '.');
980 // 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 $|y_j\rangle = 0$: 0.0625 (= 1/16).

The probability probe value of $|x_i\rangle = 16$: ≈ 0.03320 .

The example of 10 times test : The read value of $|x_i\rangle$; $R_q = 30, 26, 27, 21, 2, 20, 13, 6,$

28, 12.

The candidates of M are used by the calculated data set of M [R_q : read value, range : $2^n = 2^5 = 32$] : $R_q \rightarrow$ candidates of M ; $30 \rightarrow 13, 14, 15, 16, 17, 18, 19, 20, 21, 26, 27, 28, 29, 30, 31$; $26 \rightarrow 10, 11, 15, 16, 17, 20, 21, 22, 23, 25, 26, 27, 28, 29, 30, 31$; $27 \rightarrow 12, 13, 14, 18, 19, 20, 21, 24, 25, 26, 27, 28, 30, 31$; $21 \rightarrow 3, 6, 9, 12, 14, 15, 17, 18, 20, 21, 23, 24, 26, 27, 28, 29, 30, 31$; $2 \rightarrow 13, 14, 15, 16, 17, 18, 19, 20, 21, 26, 27, 28, 29, 30, 31$; $20 \rightarrow 8, 11, 13, 16, 18, 19, 21, 22, 25, 26, 27, 29, 30, 31$; $13 \rightarrow 5, 10, 12, 15, 17, 19, 20, 22, 23, 24, 25, 27, 28, 29, 30, 31$; $6 \rightarrow 5, 10, 11, 15, 16, 17, 20, 21, 22, 23, 25, 26, 27, 28, 29, 30, 31$; $28 \rightarrow 8, 9, 15, 16, 17, 18, 22, 23, 24, 25, 26, 27, 29, 30, 31$; $12 \rightarrow 8, 11, 13, 16, 18, 19, 21, 22, 24, 25, 26, 27, 29, 30, 31$.

When M is 16, from $16 = 2^0m_1 + 2^1m_2 + 2^2m_3 + 2^3m_4 + 2^4m_5 = 2^0 \times 0 + 2^1 \times 0 + 2^2 \times 0 + 2^3 \times 0 + 2^4 \times 1 = 2^0x_1 + 2^1x_2 + 2^2x_3 + 2^3x_4 + 2^4x_5$, $(x_1, x_2, x_3, x_4, x_5) = (m_1, m_2, m_3, m_4, m_5)$ is obtained, and it is the answer. The spikes of 16 (= M) are 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30. After all, the sum of probability is about 0.4980 ($\approx 50\%$).

Therefore, the best times of Grover iterations are 3 for $n = 4$.

5. Discussion

In the 3-SAT, when there is no periodic in it, this method is used the calculated data set of M, where it is made by $2^n/M = S$, $1 \leq [kS] \leq 2^n - 1$, $k =$ (natural number), and $[kS] =$ (kS is rounded to one decimal place.).

For example, the data set of candidates of M for R_q and $n = 5$ is the following.

$R_q \rightarrow$ candidates of M ($n = 5$) ; $1 \rightarrow 22, 23, 24, 25, 26, 27, 28, 29, 30, 31$; $2 \rightarrow 13, 14, 15, 16, 17, 18, 19, 20, 21, 26, 27, 28, 29, 30, 31$; $3 \rightarrow 10, 11, 12, 19, 20, 21, 22, 23, 24, 25, 28, 29, 30, 31$; $4 \rightarrow 8, 9, 15, 16, 17, 18, 22, 23, 24, 25, 26, 27, 29, 30, 31$; $5 \rightarrow 6, 7, 12, 13, 14, 18, 19, 20, 21, 24, 25, 26, 27, 28, 30, 31$; $6 \rightarrow 5, 10, 11, 15, 16, 17, 20, 21, 22, 23, 25, 26, 27, 28, 29, 30, 31$; $7 \rightarrow 9, 13, 14, 18, 19, 22, 23, 24, 26, 27, 28, 29, 30, 31$; $8 \rightarrow 4, 8, 12, 16, 17, 19, 20, 21, 23, 24, 25, 27, 28, 29, 31$; $9 \rightarrow 7, 11, 14, 15, 17, 18, 21, 22, 24, 25, 26, 27, 28, 29, 30, 31$; $10 \rightarrow 10, 13, 16, 19, 20, 22, 23, 25, 26, 28, 29, 30, 31$; $11 \rightarrow 3, 6, 9, 12, 14, 15, 17, 18, 20, 21, 23, 24, 26, 27, 28, 29, 30, 31$; $12 \rightarrow 8, 11, 13, 16, 18, 19, 21, 22, 24, 25, 26, 27, 29, 30, 31$; $13 \rightarrow 5, 10, 12, 15, 17, 19, 20, 22, 23, 24, 25, 27, 28, 29, 30, 31$; $14 \rightarrow 7, 9, 14, 16, 18, 20, 21, 23, 25, 27, 28, 29, 30, 31$; $15 \rightarrow 11, 13, 15, 17, 19, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31$; 16

→ 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30 ; 17 → 11, 13, 15, 17, 19, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31 ; 18 → 7, 9, 14, 16, 18, 20, 21, 23, 25, 26, 27, 28, 29, 30, 31 ; 19 → 5, 10, 12, 15, 17, 19, 20, 22, 23, 24, 25, 27, 28, 29, 30, 31 ; 20 → 8, 11, 13, 16, 18, 19, 21, 22, 25, 26, 27, 29, 30, 31 ; 21 → 3, 6, 9, 12, 14, 15, 17, 18, 20, 21, 23, 24, 26, 27, 28, 29, 30, 31 ; 22 → 10, 13, 16, 19, 20, 22, 23, 25, 26, 28, 29, 30, 31 ; 23 → 7, 11, 14, 15, 17, 18, 21, 22, 24, 25, 26, 27, 28, 29, 30, 31 ; 24 → 4, 8, 12, 16, 17, 19, 20, 21, 23, 24, 25, 27, 28, 29, 31 ; 25 → 9, 13, 14, 18, 19, 22, 23, 24, 26, 27, 28, 29, 30, 31 ; 26 → 10, 11, 15, 16, 17, 20, 21, 22, 23, 25, 26, 27, 28, 29, 30, 31 ; 27 → 12, 13, 14, 18, 19, 20, 21, 24, 25, 26, 27, 28, 30, 31 ; 28 → 8, 9, 15, 16, 17, 18, 22, 23, 24, 25, 26, 27, 29, 30, 31 ; 29 → 10, 11, 12, 19, 20, 21, 22, 23, 24, 25, 28, 29, 30, 31 ; 30 → 13, 14, 15, 16, 17, 18, 19, 20, 21, 26, 27, 28, 29, 30, 31 ; 31 → 22, 23, 24, 25, 26, 27, 28, 29, 30, 31.

For all of n , the complexity of the 3-SAT problem is able to be several times, and this method is simple and high speed.

6. Summary

The quantum algorithm for the 3-SAT problem by the QFT with the CCCNOT gate on the QCEngine, and its example are reported.

The complexity of this method is able to be several times, and this method is simple and high speed.

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, Programing 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].