Global Journal of Pure and Applied Mathematics.
ISSN 0973-1768 Volume 20, Number 1 (2024), pp. 61-69

http://www.ripublication.com

Quantum Algorithm for Clique Problem by Quantum Fourier Transform 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 clique problem by the quantum Fourier transform (= QFT) on the QCEngine, and its example are reported. When k persons of n persons form the clique, these members are decided. A complexity of a classical computation is $n!/((n-k)!k!)$. 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, clique problem, quantum Fourier transform (= QFT), QCEngine.

AMS subject classification: Primary 81-08; Secondary 68R10, 68W40.

INTRODUCTION

The clique problem has been discussed by Watanabe. [1] A quantum algorithm for the clique problem has been reported by Fujimura. [2] Still more, Fujimura discussed a

quantum algorithm for the 3-SAT problem by the quantum Fourier transform (= QFT) with the CCCNOT gate (= control Toffoli gate) on the QCEngine. [3]

According to my advanced study, when the clique problem is regarded as a special pattern of the 3-SAT problem, the complexity of the clique 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 clique problem is examined by the QFT on the QCEngine, its result is reported.

2. Clique Problem

When k of n is forming the clique, these members are decided. A complexity of a classical computation is $n! / ((n-k)!\ k!)$ Times because a number of the combinations chooses k from n, and its order is unrelated.

3. Quantum Algorithm

3-1. 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. [4-6]

3-2. From 3-SAT Problem To Clique Problem

The following conditions are assumed. (I) each value of variables x_1, x_2, x_{n-1}, 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_{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 $2^0 m_1 + 2^1 m_2 + 2^2 m_3 + \ldots + 2^{n-1} m_n$, and m_1, m_2, m_3, m_n are integers.

When a value of logical formula is 1, and a number of same value of variables (0 or 1)

is k (< n) (k is a natural number.), the k variables are regarded as the k clique.

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, … , x_n) = (0, 0, … , 0) or (1, 0, … , 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 H [1-3, 5-11] 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, $1 \leq [r\text{S}] \leq 2^n$-1, r = (natural number), and [rS] = (rS is rounded to one decimal place.).

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

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

The one-marked-term is obtained.

4. Example of Numerical Computation

For example at n = 5, it is assumed that the M = 24 (= A), logical formula : (x_3 V x_4 V x_5) & (x_3 V x_4 V ~x_5) & (~x_3 V x_4 V x_5) & (x_3 V ~x_4 V x_5) & (~x_1 V x_3 V ~x_5) & (~x_3 V ~x_4 V x_5) & (~x_3 V x_4 V ~x_5) & (~x_2 V ~x_4 V ~x_5) & (~x_3 V ~x_4 V ~x_5), each value of $x_{1\sim5}$: x_1 = x_2 = x_3 = 0, x_4 = x_5 = 1, work2 qubits = u = 9, and ancilla qubit.

Therefore, 3 variables are regarded as 3 clique.

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 query24 = 24; var query23 = 23;
```

```
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(0x4)|query.bits(0x8));
290 qc.cnot(wor2k.bits(0x2),query.bits(0x4)|query.bits(0x8)|query.bits(0x10));
300 qc.not(query.bits(0x4)|query.bits(0x8)|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(0x1)|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.cnot(work2.bits(0x80),query.bits(0x2)|query.bits(0x8)|query.bits(0x10));
```

```
470 qc.not(work2.bits(0x80));
480 qc.cnot(work2.bits(0x100),query.bits(0x4)|query.bits(0x8)|query.bits(0x10));
490 qc.not(work2.bits(0x100));
500 qc.label('a CN w2'); // ancilla CNOT work2
510 qc.cnot(ancilla.bits(0x1),work2.bits());
520 qc.label('w1 -= q'); // work1 -= query at ancilla = 1
530 work1.subtract (query,ancilla.bits(0x1));
540 qc.label('uncompute');
550 qc.cnot(ancilla.bits(0x1),work2.bits());
560 qc.not(work2.bits(0x100));
570 qc.cnot(work2.bits(0x100),query.bits(0x4)|query.bits(0x8)|query.bits(0x10));
580 qc.not(work2.bits(0x80));
590 qc.cnot(work2.bits(0x80),query.bits(0x2)|query.bits(0x8)|query.bits(0x10));
600 qc.not(query.bits(0x8)|work2.bits(0x40));
610 qc.cnot(work2.bits(0x40),query.bits(0x4)|query.bits(0x8)|query.bits(0x10));
620 qc.not(query.bits(0x8));
630 qc.not(query.bits(0x10)|work2.bits(0x20));
640 qc.cnot(work2.bits(0x20),query.bits(0x4)|query.bits(0x8)|query.bits(0x10));
650 qc.not(query.bits(0x10));
660 qc.not(query.bits(0x4)|work2.bits(0x10));
670 qc.cnot(work2.bits(0x10),query.bits(0x1)|query.bits(0x4)|query.bits(0x10));
680 qc.not(query.bits(0x4));
690 qc.not(query.bits(0x4)|query.bits(0x10)|work2.bits(0x8));
700 qc.cnot(work2.bits(0x8),query.bits(0x4)|query.bits(0x8)|query.bits(0x10));
710 qc.not(query.bits(0x4)|query.bits(0x10));
720 qc.not(query.bits(0x8)|query.bits(0x10)|work2.bits(0x4));
```

```
730 qc.cnot(work2.bits(0x4),query.bits(0x4)|query.bits(0x8)|query.bits(0x10));
740 qc.not(query.bits(0x8)|query.bits(0x10));
750 qc.not(query.bits(0x4)|query.bits(0x8)|work2.bits(0x2));
760 qc.cnot(work2.bits(0x2),query.bits(0x4)|query.bits(0x8)|query.bits(0x10));
770 qc.not(query.bits(0x4)|query.bits(0x8));
780 qc.not(query.bits(0x4)|query.bits(0x8)|query.bits(0x10)|work2.bits(0x1));
790 qc.cnot(work2.bits(0x1),query.bits(0x4)|query.bits(0x8)|query.bits(0x10));
800 qc.not(query.bits(0x4)|query.bits(0x8)|query.bits(0x10));
810 qc.label('QFT');
820 query.QFT();
830 var prob0 = 0;
840 prob0 += work1.peekProbability(work1_0); // work1_0 = 0
850 // Print output work1-Prob
860 qc.print(' Prob_work1_0: ' + prob0);
870 var prob24 = 0; var prob23 = 0;
880 prob24 += query.peekProbability(query24); prob23 += query.peekProbability(query23);
890 // Print output query-Prob
900 qc.print(' Prob_query24: ' + prob24); qc.print(' Prob_query23: ' + prob23);
910 //read
920 qc.label('Rq');
930 var b2 = query.read();
940 // Print output result
950 qc.print(' Read query = ' + b2 +'.');
960 // end
```

When this program is copied on Programming Quantum Computers https: //orelly-qc. github. io/# [free on-line quantum computation simulator QCEngine] [5], 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 $|x_i\rangle = 24$: ≈ 0.03320.

The probability probe value of $|x_i\rangle = 23$: ≈ 0.03125.

The example of 10 times test : The read value of $|x_i\rangle$; R_q = 5,3,6,21,27,5,3,31,21,31.

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

When M is 24, from $24 = 2^0m_1 + 2^1m_2 + 2^2m_3 + 2^3m_4 + 2^4m_5 = 2^0\times0 + 2^1\times0 + 2^2\times0 + 2^3\times1 + 2^4\times1 = 2^0x_1 + 2^1x_2 + 2^2x_3 + 2^3x_4 + 2^4x_5$, $(x_1, x_2, x_3, x_4, x_5) = (m_1, m_2, m_3, m_4, m_5)$ is obtained, and it is the answer. The spikes of 24 (= M) are 1, 3, 4, 5, 7, 8, 9, 11, 12, 13, 15, 16, 17, 19, 20, 21, 23, 24, 25, 27, 28, 29, 31. After all, the sum of probability is about 0.7324 (≈ 73%).

5. DISCUSSION

In the 3-SAT problem, 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, $1 \leq [r\mathrm{S}] \leq 2^n$-1, r = (natural number), and [rS] = (rS is rounded to one decimal place.).

The clique problem is regarded as same as a special pattern of the 3-SAT problem.

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

6. SUMMARY

The quantum algorithm for the clique problem by the QFT 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] Watanabe, Y., 2021, Nyumon Kogi Ryoshi Konpyuta (A Lecture of Introduction to Quantum Computer), Kodansha, Tokyo, Japan [in Japanese].

[2] Fujimura, T., 2012, "Quantum algorithm for clique problem," Glob. J. Pure Appl. Math., **8**, 175-182.

[3] Fujimura, T., 2023, "Quantum algorithm for 3-SAT problem by quantum Fourier transform with CCCNOT gate (= control Toffoli gate) on QCEngine," Glob. J. Pure Appl. Math., **19**, 641-650.

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

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

[6] 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.

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

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

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

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

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