Global Journal of Pure and Applied Mathematics.
ISSN 0973-1768 Volume 20, Number 4 (2024), pp. 747-758

https://dx.doi.org/10.37622/GJPAM/20.4.2024.747-758

Quantum Algorithm for 3-SAT Problem of 2, 3, and 4 Answers by Shor's 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 3-SAT problem of 2, 3, and 4 answers by the Shor's Fourier transform 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 $C_{u,r}$ (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, x_3, …, and x_n are variables.], and $S(u)$ is $\Sigma_{r=1 \rightarrow m}\ r \times C_{u,r}$ (x_1, x_2, x_3, … , x_n), $\mathrm{mod}(S(u)_{\max})$ of $S(u)$ [$S(u)_{\max}$ is the maximum value of $S(u)$.] is computed, next, for u, the quantum Fourier transform is done. The complexity of this Shor's Fourier transform method for the 3-SAT problem of 2, 3, and 4 answers is able to be several times.

Keywords: Quantum algorithm, 3-SAT problem, 2, 3, and 4 answers, Shor's Fourier transform, 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 the QCEngine (free on-line quantum computer simulator). [2] Fujimura discussed a quantum algorithm for the 3-SAT problem by the Shor's Fourier transform with the RAM on the QCEngine. [3]

According to my advanced study, when the Shor's Fourier transform for the 3-SAT problem of 2, 3, and 4 answers is used, the complexity of the 3-SAT problem of 2, 3, and 4 answers is able to be several times.

Therefore, because the quantum algorithm for the 3-SAT problem of 2, 3, and 4 answers is examined by the Shor's Fourier transform 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 $2^0x_1 + 2^1x_2 + 2^2x_3 + \ldots + 2^{n-1}x_n$, a number of clauses is m, an r-th clause ($1 \leq r \leq m$) is $C_{u,r}(x_1, x_2, x_3, \ldots, x_n)$, $S(u)$ is $\Sigma_{r=1 \rightarrow m}\, r \times C_{u,r}(x_1, x_2, x_3, \ldots, x_n)$, and $S(u)_{max}$ is (the maximum value of $S(u)$) = $(m + 1)m/2 = k$.

First of all, query quantum registers $|x_i\rangle[1 \leq i \leq n$. i is an integer. n is the number of variables in logical formula.], work1 quantum registers $|w_{1,j}\rangle$ $[1 \leq j \leq t$. j, and t are integers. t is a necessary number for $S(u)_{max} \leq 2^t$.], 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 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 $0 \rightarrow (m - 1)$.] is incremented in $|w_{2,p}\rangle$. In a function, $S(u) = \Sigma_{r=1 \rightarrow m}\, r \times C_{u,r}\, (x_1, x_2, x_3, \ldots, x_n)$ is computed. This operation makes entangled data base.

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

Step 6: For $|x_i\rangle$, the quantum Fourier transform (= QFT) [2-6] 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_1, P_2, P_3, and P_4 are obtained.

Step 11: From $u = P_1$, P_2, P_3, and P_4, when there are $S(u)$ is $\Sigma_{r=1 \rightarrow m}\, r \times C_{u,r}\, (x_1, x_2, x_3, \ldots, x_n) = S(u)_{max} = k$, they are answers [numbers of combination of (value of logical formula) = 1].

4. Example of Numerical Computation

4.1. 4 Answers

For example at $n = 5$, it is assumed that logical formula : $(x_3 \text{ V } x_4 \text{ V } x_5)$ & $(\sim x_1 \text{ V } x_2 \text{ V } x_3)$ & $(\sim x_1 \text{ V } x_4 \text{ V } x_5)$ & $(\sim x_2 \text{ V } x_4 \text{ V } x_5)$ & $(\sim x_2 \text{ V } x_3 \text{ V } x_5)$ & $(\sim x_2 \text{ V } x_3 \text{ V } \sim x_5)$ & $(\sim x_1 \text{ V } \sim x_3 \text{ V } \sim x_4)$ & $(\sim x_2 \text{ V } \sim x_3 \text{ V } \sim x_4)$ & $(\sim x_3 \text{ V } x_4 \text{ V } \sim x_5)$ & $(x_3 \text{ V } \sim x_4 \text{ V } \sim x_5)$ & $(\sim x_3 \text{ V } \sim x_4 \text{ V } \sim x_5)$, $(x_1, x_2, x_3, x_4, x_5)$ = [(0, 0, 1, 0, 0), (0, 0, 0, 1, 0), (0, 0, 1, 1, 0), and (0, 0, 0, 0, 1)], $m = 11$, $t = 7$, and $k = (m + 1)m/2 = 66$.

An example of program on the QCEngine is the following.

```
10 var a = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]; // RAM_a
20 var query_qubits = 5;
30 var work1_qubits = 7;
40 var work2_qubits = 8;
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 query4 = 4;
230 var query8 = 8;
240 var query12 = 12:
250 var query16 = 16;
260 var k = 66;
270 var work1_0 = 0;
280 qc.label('increment');
290 qc.not(query.bits(0x4)|query.bits(0x8)|query.bits(0x10));
300 work2.add(a[0],query.bits(0x4)|query.bits(0x8)|query.bits(0x10));
310 qc.not(query.bits(0x4)|query.bits(0x8)|query.bits(0x10));
320 qc.not(query.bits(0x2)|query.bits(0x4));
330 work2.add(a[1],query.bits(0x1)|query.bits(0x2)|query.bits(0x4));
340 qc.not(query.bits(0x2)|query.bits(0x4));
350 qc.not(query.bits(0x8)|query.bits(0x10));
360 work2.add(a[2],query.bits(0x1)|query.bits(0x8)|query.bits(0x10));
370 qc.not(query.bits(0x8)|query.bits(0x10));
380 qc.not(query.bits(0x8)|query.bits(0x10));
390 work2.add(a[3],query.bits(0x2)|query.bits(0x8)|query.bits(0x10));
400 qc.not(query.bits(0x8)|query.bits(0x10));
410 qc.not(query.bits(0x4)|query.bits(0x10));
420 work2.add(a[4],query.bits(0x2)|query.bits(0x4)|query.bits(0x10));
430 qc.not(query.bits(0x4)|query.bits(0x10));
440 qc.not(query.bits(0x4));
450 work2.add(a[5],query.bits(0x2)|query.bits(0x4)|query.bits(0x10));
```

```
460 qc.not(query.bits(0x4));
470 work2.add(a[6],query.bits(0x1)|query.bits(0x4)|query.bits(0x8));
480 work2.add(a[7],query.bits(0x2)|query.bits(0x4)|query.bits(0x8));
490 qc.not(query.bits(0x8));
500 work2.add(a[8],query.bits(0x4)|query.bits(0x8)|query.bits(0x10));
510 qc.not(query.bits(0x8));
520 qc.not(query.bits(0x4));
530 work2.add(a[9],query.bits(0x4)|query.bits(0x8)|query.bits(0x10));
540 qc.not(query.bits(0x4));
550 work2.add(a[10],query.bits(0x4)|query.bits(0x8)|query.bits(0x10));
560 qc.label('mod(' + k + ')');
570 work2.subtract(k);
580 qc.cnot(ancilla.bits(0x1), work2.bits(0x80));
590 work2.add(k,ancilla.bits(0x1));
600 work1.add(work2);
610 qc.label('uncompute');
620 work2.subtract(k,ancilla.bits(0x1));
630 qc.cnot(ancilla.bits(0x1),work2.bits(0x80));
640 work2.add(k);
650 work2.subtract(a[10],query.bits(0x4)|query.bits(0x8)|query.bits(0x10));
660 qc.not(query.bits(0x4));
670 work2.subtract(a[9],query.bits(0x4)|query.bits(0x8)|query.bits(0x10));
680 qc.not(query.bits(0x4));
690 qc.not(query.bits(0x8));
700 work2.subtract(a[8],query.bits(0x4)|query.bits(0x8)|query.bits(0x10));
710 qc.not(query.bits(0x8));
```

```
720 work2.subtract(a[7],query.bits(0x2)|query.bits(0x4)|query.bits(0x8));
730 work2.subtract(a[6],query.bits(0x1)|query.bits(0x4)|query.bits(0x8));
740 qc.not(query.bits(0x4));
750 work2.subtract(a[5],query.bits(0x2)|query.bits(0x4)|query.bits(0x10));
760 qc.not(query.bits(0x4));
770 qc.not(query.bits(0x4)|query.bits(0x10));
780 work2.subtract(a[4],query.bits(0x2)|query.bits(0x4)|query.bits(0x10));
790 qc.not(query.bits(0x4)|query.bits(0x10));
800 qc.not(query.bits(0x8)|query.bits(0x10));
810 work2.subtract(a[3],query.bits(0x2)|query.bits(0x8)|query.bits(0x10));
820 qc.not(query.bits(0x8)|query.bits(0x10));
830 qc.not(query.bits(0x8)|query.bits(0x10));
840 work2.subtract(a[2],query.bits(0x1)|query.bits(0x8)|query.bits(0x10));
850 qc.not(query.bits(0x8)|query.bits(0x10));
860 qc.not(query.bits(0x2)|query.bits(0x4));
870 work2.subtract(a[1],query.bits(0x1)|query.bits(0x2)|query.bits(0x4));
880 qc.not(query.bits(0x2)|query.bits(0x4));
890 qc.not(query.bits(0x4)|query.bits(0x8)|query.bits(0x10));
900 work2.subtract(a[0],query.bits(0x4)|query.bits(0x8)|query.bits(0x10));
910 qc.not(query.bits(0x4)|query.bits(0x8) |query.bits(0x10));
920 qc.label('QFT');
930 query.QFT();
940 var prob4 = 0;
950 var prob8 = 0;
960 var prob12 = 0;
970 var prob16 = 0;
```

```
980 prob4 += query.peekProbability(query4);
990 prob8 += query.peekProbability(query8);
1000 prob12 += query.peekProbability(query12);
1010 prob16 += query.peekProbability(query16);
1020 // Print output query-Prob
1030 qc.print(' Prob_query4: ' + prob4);
1040 qc.print(' Prob_query8: ' + prob8);
1050 qc.print(' Prob_query12: ' + prob12);
1060 qc.print(' Prob_query16: ' + prob16);
1070 var prob0 = 0;
1080 prob0 += work1.peekProbability(work1_0);
1090 // Print output work1_0-Prob
1100 qc.print(' Prob_work1_0: ' + prob0);
1110 //read
1120 qc.label('Rq');
1130 var b2 = query.read();
1140 // Print output result
1150 qc.print(' Read query = ' + b2 +'.');
1160 // 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 $|w_{1,j}\rangle = 0 : 0.12500$ (= 4/32 = 1/8).

The probability probe value of $|x_i\rangle = 4 : \approx 0.037583$.

The probability probe value of $|x_i\rangle = 8 : \approx 0.035156$.

The probability probe value of $|x_i\rangle = 12 : \approx 0.021010$.

The probability probe value of $|x_i\rangle = 16 : \approx 0.039093$.

The example of 10 times test : The read value of $|x_i\rangle$; $R_q = 5, 20, 23, 18, 4, 7, 16, 20, 1, 8$. (=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 ; $5 \rightarrow 6, 13, 19$; $20 \rightarrow 3, 5, 8, 16$; $23 \rightarrow 4, 7, 14, 18$; $18 \rightarrow 2, 5, 7, 9, 16$; $4 \rightarrow 8, 16, 24$; $7 \rightarrow 5, 9, 18, 23$; $16 \rightarrow 2, 4, 6, 8, 10, 12, 14$; $1 \rightarrow$ nothingness ; $8 \rightarrow 4, 8, 12, 16, 20$.

When u is 4 ($2^0x_1 + 2^1x_2 + 2^2x_3 + 2^3x_4 + 2^4x_5 = 2^0\times0 + 2^1\times0 + 2^2\times1 + 2^3\times0 + 2^4\times0 = 4$), 8 ($2^0\times0 + 2^1\times0 + 2^2\times0 + 2^3\times1 + 2^4\times0 = 8$), 12 ($2^0\times0 + 2^1\times0 + 2^2\times1 + 2^3\times1 + 2^4\times0 = 12$), and 16 ($2^0\times0 + 2^1\times0 + 2^2\times0 + 2^3\times0 + 2^4\times1 = 16$), the value of logical formula is 1. Therefore, they are answers.

4.2. 3 Answers

For example at $n = 5$, it is assumed that logical formula : $(x_3 \vee x_4 \vee x_5) \wedge (\sim x_1 \vee x_2 \vee x_3) \wedge (\sim x_1 \vee x_4 \vee x_5) \wedge (\sim x_2 \vee x_4 \vee x_5) \wedge (\sim x_2 \vee x_3 \vee x_5) \wedge (\sim x_2 \vee x_3 \vee \sim x_5) \wedge (\sim x_3 \vee \sim x_4 \vee x_5) \wedge (\sim x_3 \vee x_4 \vee \sim x_5) \wedge (x_3 \vee \sim x_4 \vee \sim x_5) \wedge (\sim x_3 \vee \sim x_4 \vee \sim x_5)$, $(x_1, x_2, x_3, x_4, x_5) = [(0, 0, 1, 0, 0), (0, 0, 0, 1, 0)$, and $(0, 0, 0, 0, 1)]$, $m = 10$, $t = 6$, and $k = (m + 1)m/2 = 55$.

A result of this problem is the following.

The probability probe value of $|w_{1,j}\rangle = 0 : 0.093750$ (= 3/32).

The probability probe value of $|x_i\rangle = 4 : \approx 0.042299$.

The probability probe value of $|x_i\rangle = 8 : \approx 0.025391$.

The probability probe value of $|x_i\rangle = 16 : \approx 0.021484$.

The example of 10 times test : The read value of $|x_i\rangle$; $R_q = 31, 0, 8, 4, 24, 12, 31, 9, 11, 17$. (=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 ; $31 \rightarrow$ nothingness ; $0 \rightarrow$ nothingness ; $8 \rightarrow 4, 8, 12, 16, 20$; $4 \rightarrow 8, 16, 24$; $24 \rightarrow 4, 8, 12, 16, 20$; $12 \rightarrow 3, 5, 8, 16$; $9 \rightarrow 4, 7, 14, 18$; $11 \rightarrow 3, 6, 9, 12, 15,$

17 ; 17 → 2, 4, 6, 9, 11, 13, 15.

When u is 4 ($2^0x_1 + 2^1x_2 + 2^2x_3 + 2^3x_4 + 2^4x_5 = 2^0\times0 + 2^1\times0 + 2^2\times1 + 2^3\times0 + 2^4\times0 = 4$), 8 ($2^0x_1 + 2^1x_2 + 2^2x_3 + 2^3x_4 + 2^4x_5 = 2^0\times0 + 2^1\times0 + 2^2\times0 + 2^3\times1 + 2^4\times0 = 8$), and 16 ($2^0\times0 + 2^1\times0 + 2^2\times0 + 2^3\times0 + 2^4\times1 = 16$), the value of logical formula is 1. Therefore, they are answers.

4.3. 2 Answers

For example at $n = 5$, it is assumed that logical formula : (x_3 V x_4 V x_5) & (~x_1 V x_2 V x_3) & (~x_3 V x_4 V x_5) & (~x_2 V x_3 V x_5) & (~x_2 V x_3 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_3 V ~x_4 V ~x_5), (x_1, x_2, x_3, x_4, x_5) = [(0, 0, 0, 1, 0), and (0, 0, 0, 0, 1)], $m = 9$, $t = 6$, and $k = (m + 1)m/2 = 45$.

A result of this problem is the following.

The probability probe value of $|w_{1,j}\rangle = 0$: 0.062500 (= 1/16).

The probability probe value of $|x_i\rangle = 8$: ≈ 0.023438.

The probability probe value of $|x_i\rangle = 16$: ≈ 0.011719.

The example of 10 times test : The read value of $|x_i\rangle$; R_q = 30, 29, 31, 28, 8, 3, 21, 25, 13, 10. (=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 → candidates ; 30 → 16 ; 29 → 11, 21 ; 31 → nothingness ; 28 → 8, 16, 24 ; 8 → 4, 8, 12, 16, 20 ; 3 → 11,21 ; 21 → 3, 6, 9, 12, 15, 17 ; 25 → 5, 9, 18, 23 ; 13 → 3, 5, 10, 15, 17 ; 10 → 3, 6, 10, 13, 16.

When u is 8 ($2^0x_1 + 2^1x_2 + 2^2x_3 + 2^3x_4 + 2^4x_5 = 2^0\times0 + 2^1\times0 + 2^2\times0 + 2^3\times1 + 2^4\times0 = 8$), and 16 ($2^0\times0 + 2^1\times0 + 2^2\times0 + 2^3\times0 + 2^4\times1 = 16$), the value of logical formula is 1. Therefore, they are answers.

5. Discussion

In the 3-SAT problem of 2, 3, and 4 answers, when [(the logical formula) = 1] is obtained, there are 2, 3, and 4 combinations.

5.1. 4 Answers

In the section 4-1, 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 $4N^{1/2} = 4 \cdot 2^{5/2} \approx 24$, in the Shor's Fourier transform, it is for 4, $10/3 \approx 3$, for 8, $10/5 = 2$, for 12, $10/2 = 5$, and for 16, $10/5 = 2$.

5.2. 3 Answers

In the section 4-2, 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 $3N^{1/2} = 3 \cdot 2^{5/2} \approx 18$, in the Shor's Fourier transform, it is for 4, $10/4 = 2.5 \approx 3$, for 8, $10/4 \approx 3$, and for 16, $10/4 \approx 3$.

5.3. 2 Answers

In the section 4-3, 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 $2N^{1/2} = 2 \cdot 2^{5/2} \approx 12$, in the Shor's Fourier transform, it is for 8, $10/2 = 5$, and for 16, $10/4 = 2.5 \approx 3$.

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 of 2, 3, and 4 answers by the Shor's Fourier transform 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., 2024, “Quantum algorithm for 3-SAT problem by Shor’s Fourier transform with RAM on QCEngine,” Glob. J. Pure Appl. Math., **20**, 695-703.

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

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

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

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

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