Global Journal of Pure and Applied Mathematics.
ISSN 0973-1768 Volume 20, Number 3 (2024), pp. 577-584

https://dx.doi.org/10.37622/GJPAM/20.3.2024.577-584

Quantum Algorithm for Minimum Traveling Repairman 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 minimum traveling repairman problem by the Shor's Fourier transform with the RAM on the QCEngine, and its example are reported. When a graph has n vertexes and the m edges, the shortest path from the initial vertex s passing through n vertexes that are included s is decided. A complexity of a classical computation is $(n - 1)!$ times. In the quantum algorithm by the Shor's Fourier transform with the RAM, its search is done by about (several times)$\times \log_2 N$, where N is 2^m.

Keywords: Quantum algorithm, minimum traveling repairman problem, Shor's Fourier transform, RAM, QCEngine.

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

1. Introduction

A quantum algorithm for the minimum traveling repairman problem by a numbering method was discussed by Fujimura. [1] Still more, Fujimura discussed a quantum algorithm for the traveling salesman problem by the Shor's Fourier transform with the RAM on the QCEngine. [2]

According to my advanced study, when the minimum traveling repairman problem is

regarded as a special pattern of the traveling salesman problem, the complexity of the minimum traveling repairman problem is able to be about (several times)×$\log_2 N$, where N is 2^m [m is number of edges.].

Therefore, because the quantum algorithm for the minimum traveling repairman problem is examined by the Shor's Fourier transform with the RAM on the QCEngine, its result is reported.

2. Minimum Traveling Repairman Problem

When a graph has n vertexes and m edges, the shortest path from the initial vertex s passing through n vertexes that are included s is decided. A complexity of a classical computation is $(n - 1)!$ times. [1]

3. Quantum Algorithm

It is assumed that n is number of vertexes, m is number of edges ($m = n(n - 1)/2$, and number of data qubits), and j is number of work qubits that included the sum of distances (distance is length between two vertexes.).

First of all, query quantum registers $|x_i\rangle$ [$1 \leq i \leq m$. i and m are integers. m is a number of edges.], work1 quantum registers $|w_{1,\,j}\rangle$ [$1 \leq j \leq t$. j and t are integers. t is a necessary number for the sum of distances.], work2 quantum registers $|w_{2,\,p}\rangle$ [$1 \leq p \leq t$ +1. p and t are integers. t is a necessary number for the sum of distances. +1 is a qubit for the negative integer. [3]], and ancilla quantum qubits $|a_q\rangle$ [q is a necessary number for decrement with modulus.] are prepared.

Step 1: The distance data [$d(u_i, v_i)$: i-th distance between u_i and v_i. $u_i \neq v_i$. u_i and v_i are vertexes.] are introduced to the RAM [3].

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

Step 3: The Hadamard gate H [1-8] acts on each qubit of $|x_i\rangle$. It changes them for entangled states.

Step 4: For $|x_i\rangle$, RAM [i - 1] [RAM has distance data of $0 \rightarrow (m - 1)$.] is incremented in $|w_{2,\,p}\rangle$. In a function, $F = \Sigma_{i\,=\,1\,\rightarrow\,m}\, d(u_i, v_i)x_i$ is computed, where $d(u_i, v_i)$ is i-th distance. This operation makes entangled data base.

Step 5: For $|w_{2,\,p}\rangle$, mod(k) [k is a length of a route, and at random.] is done, where mod(k) is made by subtraction and addition in this program. [3] Therefore, the subtraction and the addition are done by necessary times, 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) [2-6] 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 [3]), where the function estimate_num_spikes (spike, range) [spike: read value, range: 2^m] is used.

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

Step 11: From $P = 2^0x_1 + 2^1x_2 + 2^2x_3 + 2^3x_4 + \ldots + 2^{m-1}x_m$, when there is $k = d(u_1, v_1)x_1 + d(u_2, v_2)x_2 + \ldots + d(u_m, v_m)x_m$, it is answer [number of combination of necessary distances].

Step 12: The repeating times is $\log_2 N$ [$N = 2^m$] to obtain a minimum length k_{min}. [9]

4. Example of Numerical Computation

It is assumed that 9 (= m) distances are $d(u_1, v_1) = d(1, 2) = 1$, $d(u_2, v_2) = d(2, 3) = 1$, $d(u_3, v_3) = d(3, 4) = 1$, $d(u_4, v_4) = d(4, 5) = 1$, $d(u_5, v_5) = d(1, 3) = 2$, $d(u_6, v_6) = d(1, 4) = 2$, $d(u_7, v_7) = d(2, 4) = 2$, $d(u_8, v_8) = d(2, 5) = 2$, $d(u_9, v_9) = d(3, 5) = 2$, and the upper bound of the length is 14, and $n = 5$. Furthermore, it is assumed that the mod(k_{min}) = mod(4), and query quantum register qubits $i = m = 9$. In this example, when mod(4) is 0, P is 0, 15, 19, 21, 22, 25, 26, 28, 35, 37, 38, 41, 42, 44, 48, 63, 67, 69, 70, 73, 74, 76, 80, 95, 96, 111, 115, 117, 118, 121, 122, 124, 131, 133, 134, 137, 138, 140, 144, 159, 160, 175, 179, 181, 182, 185, 186, 188, 192, 207, 211, 213, 214, 217, 218, 220, 227, 229, 230, 233, 234, 236, 240, 255, 259, 261, 265, 266, 268, 272, 287, 288, 303, 307, 309, 310, 313, 314, 316, 320, 335, 339, 341, 342, 345, 346, 348, 355, 357, 358, 361, 362, 364, 368, 383, 384, 399, 403, 405, 406, 409, 410, 412, 419, 421, 422, 425, 426, 428, 432, 447, 451, 453, 454, 457, 458, 460, 464, 479, 480, 495, 499, 501, 502, 505, 506, and 508.

An example of program on the QCEngine is the following.

```
10 var a = [1, 1, 1, 1, 2, 2, 2, 2, 2]; // RAM_a, distance data.
20 var query_qubits = 9;
30 var work1_qubits = 4;
40 var work2_qubits = 5;
50 var ancilla_qubits = 3; // subtractions of 3 times are able.
60 qc.reset(query_qubits + work1_qubits + work2_qubits + ancilla_qubits);
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_qubits, '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 query15 = 15; // one of query
230 var k = 4; // one of length
240 var work1_0 = 0; // one of work1. one of mod(k). k = 4.
250 qc.label('increment');
260 work2.add(a[0],query.bits(0x1));
270 work2.add(a[1],query.bits(0x2));
280 work2.add(a[2],query.bits(0x4));
290 work2.add(a[3],query.bits(0x8));
300 work2.add(a[4],query.bits(0x10));
310 work2.add(a[5],query.bits(0x20));
320 work2.add(a[6],query.bits(0x40));
330 work2.add(a[7],query.bits(0x80));
340 work2.add(a[8],query.bits(0x100));
350 qc.label('mod(' + k + ')');
360 work2.subtract(k);
370 qc.cnot(ancilla.bits(0x1),work2.bits(0x10));
380 work2.add(k, ancilla.bits(0x1));
390 work2.subtract(k);
400 qc.cnot(ancilla.bits(0x2),work2.bits(0x10));
410 work2.add(k, ancilla.bits(0x2));
```

```
420 work2.subtract(k);
430 qc.cnot(ancilla.bits(0x4),work2.bits(0x10));
440 work2.add(k, ancilla.bits(0x4));
450 work1.add(work2);
460 qc.label('uncompute');
470 work2.subtract(k,ancilla.bits(0x4));
480 qc.cnot(ancilla.bits(0x4),work2.bits(0x10));
490 work2.add(k);
500 work2.subtract(k,ancilla.bits(0x2));
510 qc.cnot(ancilla.bits(0x2),work2.bits(0x10));
520 work2.add(k);
530 work2.subtract(k,ancilla.bits(0x1));
540 qc.cnot(ancilla.bits(0x1),work2.bits(0x10));
550 work2.add(k);
560 work2.subtract(a[8],query.bits(0x100));
570 work2.subtract(a[7],query.bits(0x80));
580 work2.subtract(a[6],query.bits(0x40));
590 work2.subtract(a[5],query.bits(0x20));
600 work2.subtract(a[4],query.bits(0x10));
610 work2.subtract(a[3],query.bits(0x8));
620 work2.subtract(a[2],query.bits(0x4));
630 work2.subtract(a[1],query.bits(0x2));
640 work2.subtract(a[0],query.bits(0x1));
650 qc.label('QFT');
660 query.QFT();
670 var prob15 = 0;
680 prob15 += query.peekProbability(query15);
690 // Print output query-Prob
700 qc.print(' Prob_query15: ' + prob15);
710 var prob0 = 0;
720 prob0 += work1.peekProbability(work1_0); // work1_0 = 0
```

```
730 // Print output work1-Prob
740 qc.print(' Prob_work1_0: ' + prob0);
750 // read
760 qc.label('Rq');
770 var b2 = query.read();
780 // Print output result
790 qc.print(' Read query = ' + b2 + '.');
800 // end
```

When this program is copied on Programming Quantum Computers https: //orelly-qc. github. io/# [free on-line quantum computation simulator QCEngine] [3], you can run it. [Caution!: Please delate the line numbers.]

A result of this program is the following.

The probe value of $|w_{1,j}\rangle = 0 : \approx 0.2500$.

The probe value of $|x_i\rangle = 15 : \approx 0.0002314$.

The example of 10 times test: The read value of $|x_i\rangle$ = 297, 224, 77, 0, 485, 245, 69, 0, 96, 501. (= spike)

The candidates of number of spikes are estimated by the function [the function estimate_num_spikes (spike, range) [spike: read value, range: $2^m = 2^9 = 512$]]:

297 → 2, 5, 7, 12, 19, 31, 50, 100, 131, 181, 231, 281 ; 224 → 2, 5, 7, 9, 16, 32, 48, 64, 80, 96, 112, 128, 144, 160, 176, 192, 208, 224, 240, 256, 272 ; 77 → 7, 13, 20, 40, 60, 73, 93, 113, 133, 266, 379 ; 0 → nothingness ; 485 → 19, 38, 57, 76, 95, 114, 133, 152, 171, 190, 209, 228, 247, 265, 284, 303, 322, 341, 360, 379, 398, 417, 436, 455, 474 ; 245 → 2, 4, 6, 8, 10, 13, 15, 17, 19, 21, 23, 46, 69, 92, 94, 117, 140, 163 ; 69 → 7, 15, 30, 37, 52, 89, 141, 282, 371 ; 0 → nothingness ; 96 → 5, 11, 16, 32, 48, 64, 80, 96, 112, 128, 144, 160, 176, 192, 208, 224, 240, 256, 272, 288, 304, 320, 336, 352, 368, 384, 400 ; 501 → 47, 93, 186, 279, 326, 419.

When P is 15, $2^0x_1 + 2^1x_2 + 2^2x_3 + 2^3x_4 + 2^4x_5 + 2^5x_6 + 2^6x_7 + 2^7x_8 + 2^8x_9$:

$2^0\times1 + 2^1\times1 + 2^2\times1 + 2^3\times1 + 2^4\times0 + 2^5\times0 + 2^6\times0 + 2^7\times0 + 2^8\times0 = 15$.

There is $d(1, 2)x_1 + d(2, 3)x_2 + d(3, 4)x_3 + d(4, 5)x_4 + d(1, 3)x_5 + d(1, 4)x_6 + d(2, 4)x_7 + d(2, 5)x_8 + d(3, 5)x_9$:

$1\times1 + 1\times1 + 1\times1 + 1\times1 + 2\times0 + 2\times0 + 2\times0 + 2\times0 + 2\times0 = 4$ (= k_{min}).

The repeating times is $\log_2 2^9 = 9$ [$N = 2^m$, $m = 9$] to obtain a minimum length k_{min}.

5. Discussion

In the minimum traveling repairman problem, when the shortest value is obtained, there is only one combination of distances (distance is length between two vertexes).

Therefore, the search is difficult.

In section 4, n vertexes' graph has $m = 9$ edges. And then, in m edges, the shortest combination is selected. When N is 2^m, in the Grover's method, the complexity is $N^{1/2}\log_2 N$, in the Shor's Fourier transform, it is (several times)$\times\log_2 N$.

In $n = 5$ and $m = 9$, $N^{1/2}\log_2 N = 2^{9/2}\log_2 2^9 \approx 23\times 9 = 207$, and (several times)$\times\log_2 N \leq (10/2)\times 9 = 45$.

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 minimum traveling repairman 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)$\times\log_2 N$, where N is 2^m [m is number of edges.].

I will apply this method for other problems.

References

[1] Fujimura, T., 2019, "Quantum algorithm for minimum traveling repairman problem by numbering method," Adva. Theo. Appl. Math., **14**, 73-80.

[2] Fujimura, T., 2024, "Quantum algorithm for traveling salesman problem by Shor's Fourier transform with RAM on QCEngine," Glob. J. Pure Appl. Math., **20**, 143-150.

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

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

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

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

[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.

[9] Durr, C., and Hoyer, P., 1996, “A quantum for finding the minimum,” [Online], Available: http: //arXiv. org /quant-ph /arXiv: quant-ph /9607014v2.