Global Journal of Pure and Applied Mathematics.
ISSN 0973-1768 Volume 20, Number 3 (2024), pp. 519-526

https://dx.doi.org/10.37622/GJPAM/20.3.2024.519-526

Quantum Algorithm for Maximum Integer M-Dimensional Knapsack 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 maximum integer m-dimensional knapsack problem by the Shor's Fourier transform with the RAM on the QCEngine, and its example are reported. When an optimal combination of n pieces of different weight luggage packed into each knapsack that a weight T_r [$1 \leq r \leq m$. r and m are integers.] can be put is requested, a complexity of a classical computation is $(2^n - 1)^m$. The complexity becomes about m×(several times) by the Shor's Fourier transform with the RAM on the QCEngine.

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

AMS subject classification: Primary 81-08; Secondary 68R05, 68W40.

1. Introduction

A quantum algorithm for the maximum integer m-dimensional knapsack problem by a numbering method was discussed by Fujimura. [1] Still more, Fujimura discussed a quantum algorithm for the knapsack problem by the Shor's Fourier transform with the RAM on the QCEngine. [2]

According to my advanced study, when the maximum integer m-dimensional knapsack problem is regarded as a special pattern of the knapsack problem, the

complexity of the maximum integer m-dimensional knapsack problem is able to be m×(several times).

Therefore, the quantum algorithm for the maximum integer m-dimensional knapsack problem is examined by the Shor's Fourier transform with the RAM on the QCEngine, its result is reported.

2. Maximum Integer M-dimensional Knapsack Problem

As for n pieces of different weight luggage, the maximum integer m-dimensional knapsack problem requests an optimal combination of the luggage packed into each knapsack that a weight T_r [$1 \leq r \leq m$. r and m are integers.] is assumed to be an upper bound. [1]

3. Quantum Algorithm

It is assumed that n pieces of luggage and m knapsacks, where each weight of the luggage is T_r [$1 \leq r \leq m$. r and m are integers.] or less, and the maximum storage weight of the m knapsacks are T_r, respectively. When each weight of the luggage is u_1, u_2, … , u_n, and coefficients in which 0 or 1 are taken are x_1, x_2, … , and x_n, a sum of weights becomes $u_1x_1 + u_2x_2 + \ldots + u_nx_n$. Still more, j is number of weight qubits that included the sum of weights.

First of all, query quantum registers $|x_i\rangle$ [$1 \leq i \leq n$. i and n are integers. n is a number of luggage.], 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 weight data [$u_i : 1 \leq i \leq n$. i and n are integers. n is a number of luggage.] 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 → (n - 1). They are $u_1 \rightarrow u_n$.] is incremented in $|w_{2,p}\rangle$. In a function, $F = \Sigma_{i=1 \rightarrow n} m_i x_i$ is computed, where u_i is i-th weight. This operation makes entangled data base.

Step 5: For $|w_{2,p}\rangle$, mod (T_r)[T_r is the upper bound weight of r-th knapsack.] is done, where mod(T_r) 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^n] is used.

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

Step 11: From $P = 2^0x_1 + 2^1x_2 + 2^2x_3 + 2^3x_4 + \ldots + 2^{n-1}x_n$, when there is $T_r = u_1x_1 + u_2x_2 + \ldots + u_nx_n$, it is answer [number of combination of necessary luggages].

Step 12: Next, a value of T_i is changed, and then step 1 to step 11 are repeated to obtain answers for m knapsacks.

4. Example of Numerical Computation

It is assumed that 5 (= n) pieces of luggage of weight are $u_1 = 3$, $u_2 = 2$, $u_3 = 7$, $u_4 = 10$, $u_5 = 13$, and the upper bounds of weight of the knapsacks are $T_1 = 19$, $T_2 = 16$, $T_3 = 17$, respectively, and m is 3.

Furthermore, it is assumed that the mod(T_1) = mod(19), mod(T_2) = mod(16), mod(T_3) = mod(17), t = 6 (2^6-1 = 63. Because, total sum is $\Sigma_{i=1 \rightarrow n} u_i$ = 35.), and query quantum register qubits n = 5. In this example, when mod(19) is 0, P is 0, and 14, when mod(16) is 0, P is 0, 17, and 30, and when mod(17) is 0, P is 0, and 12.

An example of program on the QCEngine is the following.

```
10 var a = [3,2,7,10,13]; // RAM_a, weight data.
20 var query_qubits = 5;
30 var work1_qubits = 6;
40 var work2_qubits = 7;
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 query14 = 14; // one of query
230 var T1 = 19; // upper bound of knapsack (i = 1)
240 var work1_0 = 0; // one of work1. one of mod(T1). T1 = 19.
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 qc.label('mod(' + T1 + ')');
320 work2.subtract(T1);
330 qc.cnot(ancilla.bits(0x1),work2.bits(0x40));
340 work2.add(T1, ancilla.bits(0x1));
350 work2.subtract(T1);
360 qc.cnot(ancilla.bits(0x2),work2.bits(0x40));
370 work2.add(T1, ancilla.bits(0x2));
380 work2.subtract(T1);
390 qc.cnot(ancilla.bits(0x4),work2.bits(0x40));
400 work2.add(T1, ancilla.bits(0x4));
410 work1.add(work2);
420 qc.label('uncompute');
430 work2.subtract(T1,ancilla.bits(0x4));
```

```
440 qc.cnot(ancilla.bits(0x4),work2.bits(0x40));
450 work2.add(T1);
460 work2.subtract(T1,ancilla.bits(0x2));
470 qc.cnot(ancilla.bits(0x2),work2.bits(0x40));
480 work2.add(T1);
490 work2.subtract(T1,ancilla.bits(0x1));
500 qc.cnot(ancilla.bits(0x1),work2.bits(0x40));
510 work2.add(T1);
520 work2.subtract(a[4],query.bits(0x10));
530 work2.subtract(a[3],query.bits(0x8));
540 work2.subtract(a[2],query.bits(0x4));
550 work2.subtract(a[1],query.bits(0x2));
560 work2.subtract(a[0],query.bits(0x1));
570 qc.label('QFT');
580 query.QFT();
590 var prob14 = 0;
600 prob14 += query.peekProbability(query14);
610 // Print output query-Prob
620 qc.print(' Prob_query14: ' + prob14);
630 var prob0 = 0;
640 prob0 += work1.peekProbability(work1_0); // work1_0 = 0
650 // Print output work1-Prob
660 qc.print(' Prob_work1_0: ' + prob0);
670 // read
680 qc.label('Rq');
690 var b2 = query.read();
700 // Print output result
710 qc.print(' Read query = ' + b2 + '.');
720 // 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 $|w_{1,j}\rangle = 0 : \approx 0.06250$.

The probe value of $|x_i\rangle = 14 : \approx 0.04399$.

The example of 10 times test: The read value of $|x_i\rangle$ = 14, 5, 12, 16, 2, 11, 20, 10, 30, 23. (= spike)

The candidates of number of spikes are estimated by the function [the function estimate_num_spikes (spike, range) [spike: read value, range: $2^n = 2^5 = 32$]]:

14 → 2, 5, 7, 16 ; 5 → 6, 13, 19 ; 12 → 3, 5, 8, 16 ; 16 → 2, 4, 6, 8, 10, 12, 14 ; 2 → 16 ; 11 → 3, 6, 9, 12, 15, 17 ; 20 → 3, 5, 8, 16 ; 10 → 3, 6, 10, 13, 16 ; 30 → 16 ; 23 → 4, 7, 14, 18.

When P is 14, $2^0x_1 + 2^1x_2 + 2^2x_3 + 2^3x_4 + 2^4x_5$:

$2^0\times0 + 2^1\times1 + 2^2\times1 + 2^3\times1 + 2^4\times0 = 14$.

There is $u_1x_1 + u_2x_2 + u_3x_3 + u_4x_4 + u_5x_5$:

$3\times0 + 2\times1 + 7\times1 + 10\times1 + 13\times0 = 19$ (= T_1).

Next, a result of mod(16) is the following.

The probe value of $|w_{1,j}\rangle = 0 : \approx 0.09375$.

The probe value of $|x_i\rangle = 17 : \approx 0.03989$.

The example of 10 times test: The read value of $|x_i\rangle$ = 13, 29, 19, 28, 23, 11, 13, 0, 0, 5. (= spike)

The candidates of number of spikes are estimated by the function [the function estimate_num_spikes (spike, range) [spike: read value, range: $2^n = 2^5 = 32$]]:

13 → 3, 5, 10, 15, 17 ; 29 → 11, 21 ; 19 → 3, 5, 10, 15, 17 ; 28 → 8, 16, 24 ; 23 → 4, 7, 14, 18 ; 11 → 3, 6, 9, 12, 15, 17 ; 13 → 3, 5, 10, 15, 17 ; 0 → nothingness ; 0 → nothingness ; 5 → 6, 13, 19.

When P is 17, $2^0x_1 + 2^1x_2 + 2^2x_3 + 2^3x_4 + 2^4x_5$:

$2^0\times1 + 2^1\times0 + 2^2\times0 + 2^3\times0 + 2^4\times1 = 17$.

There is $u_1x_1 + u_2x_2 + u_3x_3 + u_4x_4 + u_5x_5$:

$3\times1 + 2\times0 + 7\times0 + 10\times0 + 13\times1 = 16$ (= T_2).

Last, a result of mod(17) is the following.

The probe value of $|w_{1,j}\rangle = 0 : \approx 0.06250$.

The probe value of $|x_i\rangle = 12 : \approx 0.02253$.

The example of 10 times test: The read value of $|x_i\rangle$ = 5, 5, 9, 26, 8, 8, 11, 5, 22, 5. (= spike)

The candidates of number of spikes are estimated by the function [the function estimate_num_spikes (spike, range) [spike: read value, range: $2^n = 2^5 = 32$]]:

5 → 6, 13, 19 ; 5 → 6, 13, 19 ; 9 → 4, 7, 14, 18 ; 26 → 5, 11, 16 ; 8 → 4, 8, 12, 16, 20 ; 8 → 4, 8, 12, 16, 20 ; 11 → 3, 6, 9, 12, 15, 17 ; 5 → 6, 13, 19 ; 22 → 3, 6, 10, 13, 16 ; 5 → 6, 13, 19.

When P is 12, $2^0x_1 + 2^1x_2 + 2^2x_3 + 2^3x_4 + 2^4x_5$:

$2^0\times0 + 2^1\times0 + 2^2\times1 + 2^3\times1 + 2^4\times0 = 12$.

There is $u_1x_1 + u_2x_2 + u_3x_3 + u_4x_4 + u_5x_5$:

$3\times0 + 2\times0 + 7\times1 + 10\times1 + 13\times0 = 17$ (= T_3).

After all, T_1 = 19 is (2, 7, 10), T_2 = 16 is (3, 13), and T_3 = 17 is (7, 10),

5. Discussion

In the knapsack problem, there are many combinations of luggages to obtain a value. In the maximum integer m-dimensional knapsack problem, when there are many knapsacks, the search is difficult.

In section 4, the number of knapsack are 3 (= m). In the Grover's method, the complexity is about $(2^n\text{-}1)^{m/2} = (2^5\text{-}1)^{3/2} = 31^{1/2} \approx 174$. In the Shor's Fourier transform, it is m×(several times) ≈ 3×10/3 = 10.

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 maximum integer m-dimensional knapsack problem by the Shor's Fourier transform with the RAM on the QCEngine, and its example are reported.

The complexity of this method is m×(several times).

I will apply this method for other problems.

References

[1] Fujimura, T., 2015, "Quantum algorithm for maximum integer m-dimensional knapsack problem by numbering method," Glob. J. Pure Appl. Math., **11**, 2401-2406.

[2] Fujimura, T., 2023, “Quantum algorithm for knapsack problem by Shor’s Fourier transform with RAM on QCEngine,” Glob. J. Pure Appl. Math., **19**, 547-554.

[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.