Global Journal of Pure and Applied Mathematics.
ISSN 0973-1768 Volume 19, Number 1 (2023), pp. 23-29

http://www.ripublication.com/gjpam.htm

Quantum Algorithm for Knapsack Problem by Usual Grover Iteration with *Z*-Axis-Rotation (180 degrees) 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 knapsack problem by the usual Grover iteration with *z*-axis- rotation (180 degrees) on the QCEngine, and its example are reported. In this method, there is no using QRAM, but only RAM is used. The times of iterations are about $(\pi/4)(2^n/m)^{1/2}$, where n is a number of qubits, and m is a number of marked terms. This method is simple and powerful.

Keywords: Quantum algorithm, knapsack problem, usual Grover iteration, *z*-axis-rotation (180 degrees), QCEngine, RAM.

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

1. Introduction

The knapsack problem was discussed by Takeuchi for the complexity. [1] The quantum algorithm for the knapsack problem was reported by Fujimura with the usual Grover method. [2] In this time, I used the QCEngine (Quantum Computer Simulator [3]) with RAM for this problem. This method is simple and powerful.

Therefore, because the quantum algorithm for the knapsack problem is examined by the usual Grover iteration with *z*-axis-rotation (180 degrees) on the QCEngine, its result is reported.

2. Knapsack Problem

As for n pieces of different weight luggage, the knapsack problem requests the best combination of the luggage packed into the knapsack that a weight k is assumed to be an upper bound. [1, 2]

When weights of the n pieces of luggage are assumed $x_1, x_2, \dots, x_n$, and coefficients in which 0 or 1 are taken are $m_1, m_2, \dots, m_n$, a sum of weights becomes $m_1x_1 + m_2x_2 + \dots + m_nx_n$.

It can be said from the above-mentioned fact the knapsack problem is a problem of requesting the best combination of 0 and 1 of $m_1, m_2, \dots, m_n$ in the upper bound weight k. [2]

3. Quantum Algorithm

It is assumed that n is number of data qubits (= number of luggage), and j is number of weight qubits that included the sum of weights.

First of all, query quantum registers (= query registers) $|x_i\rangle$ [$1 \le i \le n$. i and n are integers. n is a number of luggage.] and weight quantum registers (= weight registers) $|w_j\rangle$ [$1 \le j \le t$. j and t are integers. t is a necessary number for the sum of weight.] are prepared.

Step 1: The weight data [m_p : $p = 1 \rightarrow n$. p is an integer.] are introduced to RAM [3].

Step 2: Each qubit of $|x_i\rangle$ and $|w_j\rangle$ is set $|0\rangle$.

Step 3: The Hadamard gate $\boxed{H}$ [1-6] 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 weight data of $0 \rightarrow (n - 1)$. They are $m_1 \rightarrow m_n$.] is incremented in $|w_j\rangle$. In a function, $F = \Sigma_{i=1 \rightarrow n}\, m_i\, x_i$ is computed, where m_i is weight. This operation makes entangled data base.

Step 5: For $|w_j\rangle$, the flip [~ marked term 1 : marked term 1 is k.] is done.

Step 6: The uncompute is done.

Step 7: For $|x_i\rangle$, the Grover iteration is done.

Step 8: For $|w_j\rangle$ and $|x_i\rangle$, the probes are done.

Step 9: Step 4 → 8 are returned by about $(\pi/4)(2^n/m)^{1/2}$ times [3] [m is number of marked term 1.].

Step 10: For $|w_j\rangle$ and $|x_i\rangle$, the reads are done.

The read of $|w_j\rangle$ is 0, and the read of $|x_i\rangle$ is marked term 2 [= answer : number of combination of necessary luggage].

4. Example of Numerical Computation

It is assumed that 8 (= n) pieces of luggage of weight are m_1 = 13kg, m_2 = 8kg, m_3 =

3kg, m_4 = 6kg, m_5 = 15kg, m_6 = 2kg, m_7 = 4kg, m_8 = 5kg, and the upper bound of the weight of the knapsack is k = 52kg. Furthermore, it is assumed that the marked term 1 = 52 (= k), the marked term 2 = 191 (= answer), t = 6 ($2^6 - 1 = 63$. Because, total sum is $\Sigma_{p=1\rightarrow n}\, m_p = 56$.), and Grover iterations = 13 [$T_{\mathrm{best}} \approx (\pi/4)(2^n/m)^{1/2} \approx (3.14/4)(2^8/1)^{1/2} \approx 13$], theta = 180 degrees by z-axis, and query register qubits n = 8.

An example of program on the QCEngine is the following.

```
var a = [13,8,3,6,15,2,4,5];
var query_qubits = 8;
var weight_qubits = 6;
qc.reset(query_qubits + weight_qubits);
var query = qint.new(query_qubits, 'query');
var weight = qint.new(weight_qubits, 'weight');
qc.label('set q');
query.write(0);
query.hadamard();
qc.label(' ');
qc.label('set w');
weight.write(0);
qc.print('RAM before increment: '+a+'\n');
var theta = 180; // Rotation by z-axis (degrees)
var marked_term2 = 191;
var query191 = 191; // one of query
var marked_term1 = 52;
var weight0 = 0; // one of weight
var number_of_iterations = 13;
for (var i = 0; i < number_of_iterations; ++i)
{
qc.label('increment');
weight.add(a[0],query.bits(0x1));
weight.add(a[1],query.bits(0x2));
weight.add(a[2],query.bits(0x4));
weight.add(a[3],query.bits(0x8));
weight.add(a[4],query.bits(0x10));
weight.add(a[5],query.bits(0x20));
weight.add(a[6],query.bits(0x40));
weight.add(a[7],query.bits(0x80));
qc.label('flip');
weight.not(~marked_term1);
```

```
weight.cphase(theta);
weight.not(~marked_term1);
qc.label('uncompute');
weight.subtract(a[7],query.bits(0x80));
weight.subtract(a[6],query.bits(0x40));
weight.subtract(a[5],query.bits(0x20));
weight.subtract(a[4],query.bits(0x10));
weight.subtract(a[3],query.bits(0x8));
weight.subtract(a[2],query.bits(0x4));
weight.subtract(a[1],query.bits(0x2));
weight.subtract(a[0],query.bits(0x1));
qc.label('Grover');
query.hadamard();
query.not();
query.cphase(theta);
query.not();
query.hadamard();
var prob191 = 0;
prob191 += query.peekProbability(query191);
// Print output query-Probs
qc.print(
' Prob_query191: ' + prob191
);
var prob256 = 0;
prob256 += weight.peekProbability(weight0);
// Print output weight-Probs
qc.print(' Prob_weight256: ' + prob256
);
}
//read
qc.label('Rw');
var b1 = weight.read();
qc.label('Rq');
var b2 = query.read();
// Print output result
qc.print(' Read weight = ' + b1 +
',' +' Read query = ' + b2 +
```

```
'.');
//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 $|w_j\rangle = 0$: 1.0000 [$T = 1 \rightarrow 13$].

The probe value of $|x_i\rangle = 191$: $T = 1$; 0.0348, $T = 2$; 0.0946, $T = 3$; 0.1797, $T = 4$; 0.2847, $T = 5$; 0.4032, $T = 6$; 0.5276, $T = 7$; 0.6503, $T = 8$; 0.7637, $T = 9$; 0.8607, $T = 10$; 0.9352, $T = 11$; 0.9826, $T = 12$; 0.9999 (≈ 1), $T = 13$; 0.9862.

The read of $|w_j\rangle = 0$.

The read of $|x_i\rangle = 191$.

Finished in 0.522 seconds.

Therefore, the best times of Grover iterations are 12 for $n = 8$.

For $n = 6$:

RAM = [13, 8, 3, 6, 15, 2], $T_{\text{best}} \approx (\pi/4)(2^n/m)^{1/2} \approx (3.14/4)(2^6/1)^{1/2} \approx 6$.

The marked term 1 = 20 (= k), the marked term 2 = 52 (= answer).

The probe value of $|w_j\rangle = 0$: 1.0000 [$T = 1 \rightarrow 6$].

The probe value of $|x_i\rangle = 52$: $T = 1$; 0.1348, $T = 2$; 0.3439, $T = 3$; 0.5914, $T = 4$; 0.8164, $T = 5$; 0.9635, $T = 6$; 0.9966.

The read of $|w_j\rangle = 0$.

The read of $|x_i\rangle = 52$.

Therefore, the best times of Grover iterations are 6 for $n = 6$.

For $n = 4$:

RAM = [4, 3, 5, 1], $T_{\text{best}} \approx (\pi/4)(2^n/m)^{1/2} \approx (3.14/4)(2^4/1)^{1/2} \approx 3$.

The marked term 1 = 10 (= k), the marked term 2 = 13 (= answer).

The probe value of $|w_j\rangle = 0$: 1.0000 [$T = 1 \rightarrow 3$].

The probe value of $|x_i\rangle = 13$: $T = 1$; 0.4727, $T = 2$; 0.9084, $T = 3$; 0.9613.

The read of $|w_j\rangle = 0$.

The read of $|x_i\rangle = 13$.

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

For $n = 2$:

RAM = [2, 1], $T_{\text{best}} \approx (\pi/4)(2^n/m)^{1/2} \approx (3.14/4)(2^2/1)^{1/2} \approx 2$.

The marked term 1 = 2 (= k), the marked term 2 = 1 (= answer).

The probe value of $|w_j\rangle = 0$: 1.0000 [$T = 1 \rightarrow 2$].

The probe value of $|x_i\rangle = 1$: $T = 1$; 1.0000, $T = 2$; 0.2500.

The read of $|w_j\rangle = 0$. [$T = 1$].

The read of $|x_i\rangle = 1$. [$T = 1$].

Therefore, the best time of Grover iteration is 1 for $n = 2$.

For $n = 1$:

RAM = [1], $T_{\text{best}} \approx (\pi/4)(2^n/m)^{1/2} \approx (3.14/4)(2^1/1)^{1/2} \approx 1$.

The marked term 1 = 1 (= k), the marked term 2 = 1 (= answer).

The probe value of $|w_j\rangle = 0$: 1.0000 [$T = 1$].

The probe value of $|x_i\rangle = 1$: $T = 1$; 0.5000.

The read of $|w_j\rangle = 0$.

The read of $|x_i\rangle = 1$ or 0.

Therefore, the best times of Grover iterations are 2 or 1 for $n = 1$.

5. Discussion

For $n = 1$, this problem is the section of one in two data [1 or 0]. Therefore, it is inevitable.

For $n = 2$, this problem is the section of one in four data [3, 2, 1 or 0]. In usual Grover method, one time mechanism is the following.

It is assumed that the number of data is N, the value of data of $N/4$ is R, and values of data of $3N/4$ are the others. When the probability amplitudes of data of R are marked a minus, the mean of probability amplitudes becomes $(N^{-1/2}(3N/4) - N^{-1/2}(N/4))/N = (1/2)N^{-1/2}$.

When the inversion about mean is practiced, the probability amplitudes of data of R are $-(-N^{-1/2}) + (1/2)N^{-1/2}\times 2 = 2N^{-1/2}$,

and the probability amplitude of data of others are $N^{-1/2} - (N^{-1/2} - (1/2)N^{-1/2}) \times 2 = 0$.

Therefore, the sum of square of probability amplitude is $(2N^{-1/2})^2 (1/4)N + 0^2(3/4)N = 1 + 0 = 1$.

For $n = 4$, 6 or 8, the times of usual Grover iterations are about $T_{\text{best}} \approx (\pi/4)(2^n/m)^{1/2}$.

For over $n = 8$, it is assumed that above mention is true.

6. Summary

Only RAM is used, and the times of Grover iterations are about $(\pi/4)(2^n/m)^{1/2}$ [n is a number of query qubits, and m is a number of marked terms.]. This method is simple and powerful.

I will apply this method for other problems.

References

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

[2] Fujimura, T., 2010, "Quantum algorithm for knapsack problem," Glob. J. Pure Appl. Math., **6**, 263-266.

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

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

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

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