# Exploring Knapsack problem with VQE, QAOA using Qiskit

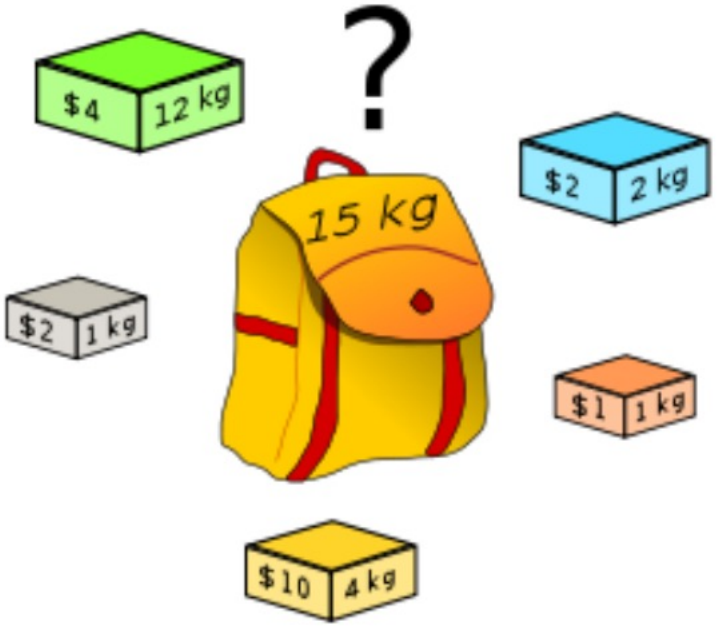
2021 quantum hackathon korea

*2021.06.30-2021.07.01*

**Team : EQ**

**Teammate : Juon Kim @rkfqns13**

# Knapsack Problem



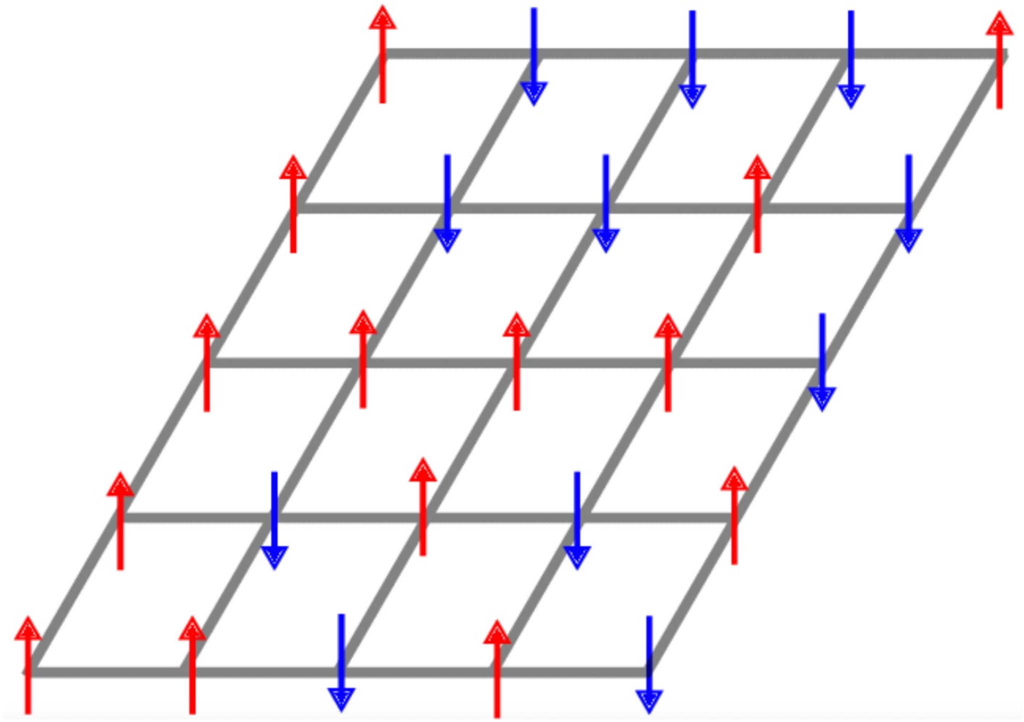

- Knapsack problem : a constrained combinatorial optimization problem that refers to the general problem of packing a knapsack with the most valuable items without exceeding its weight limit
- NP-complete problem

EX)

- values = [120, 54, 28, 49]
- weights = [13, 6, 15, 9]
- max weight = 32

-> Max value : 223, Set = [1, 1, 0, 1]

# Knapsack Problem to Ising model



- Ising model : quadratic model with binary variables without restrictions

$$H = H_A + H_B$$

$$H_A = A \sum_{j=1}^{m} \left[ b_j - \sum_{i=1}^{N} S_{ji} x_i \right]^2$$

$$H_B = -B \sum_{i=1}^{N} c_i x_i .$$

# VQE & QAOA

- Variational Quantum Eigensolver algorithm(VQE)

: a hybrid algorithm that uses a variational technique and interleaves quantum and classical computations

in order to find the minimum eigenvalue of the Hamiltonian H of a given system

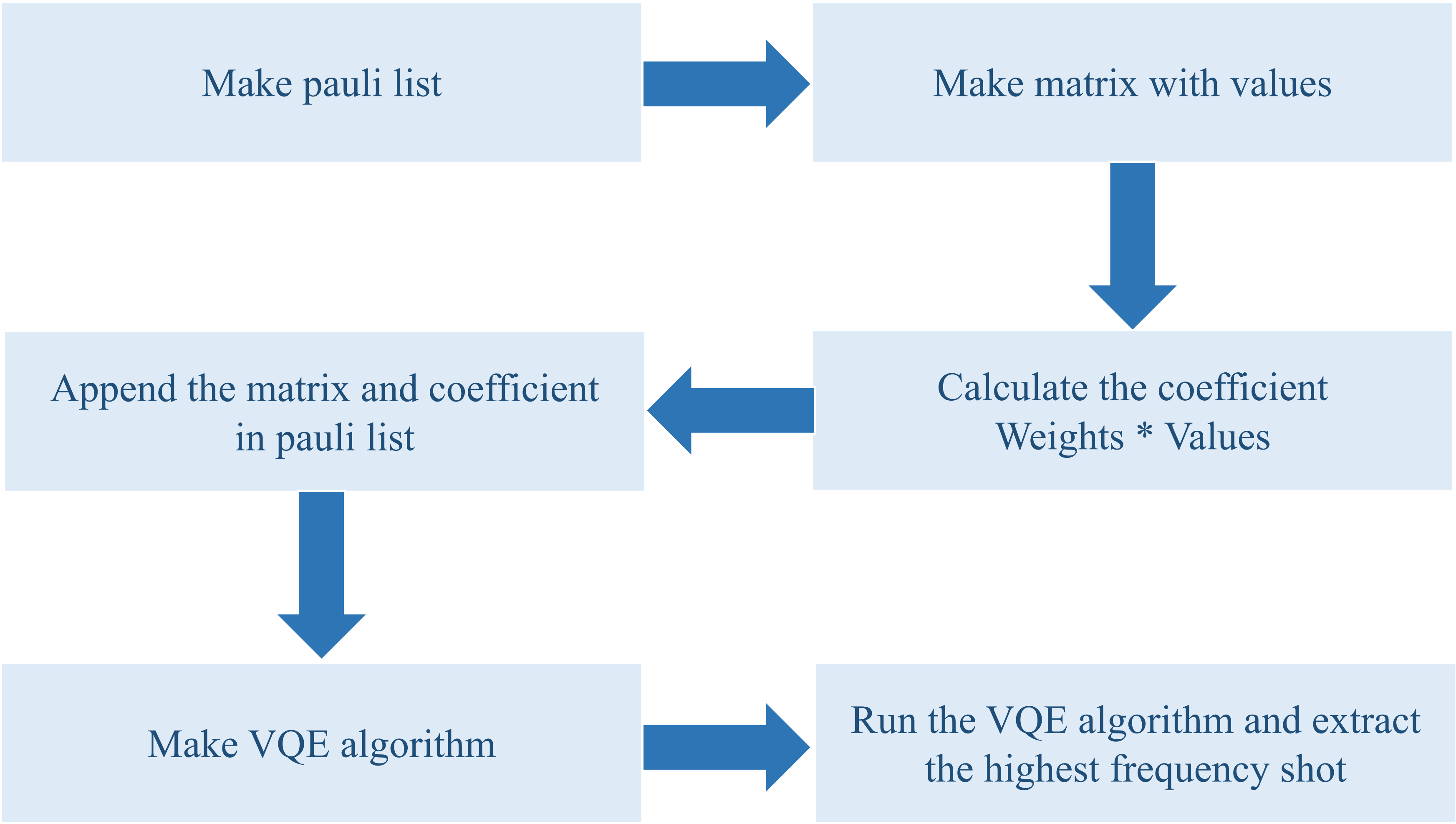
- Quantum Approximate Optimization Algorithm(QAOA)

: a general technique that can be used to find approximate solutions to combinatorial optimization

problems, in particular problems that can be cast as searching for an optimal bitstring

-> Most popular hybrid quantum-classical algorithms

# Knapsack Problem with VQE



# Knapsack Problem with VQE

1. Make the get_knapsack_operator function

- Make pauli list
- Declare variable shift (angle of ry rotation)
- Coefficient = weights * values
- Make two zero matrices with values and apply not gate to one matrix
- Append the coefficient and two matrices in pauli list
- Shift value subtract the coefficient value

→ Repeat the 4 lines below for every items

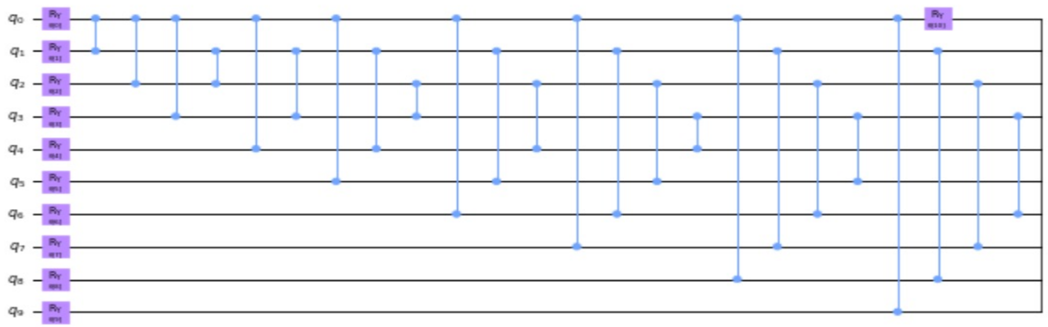
- Apply get_knapsack_operator function with values, weights, and w_max

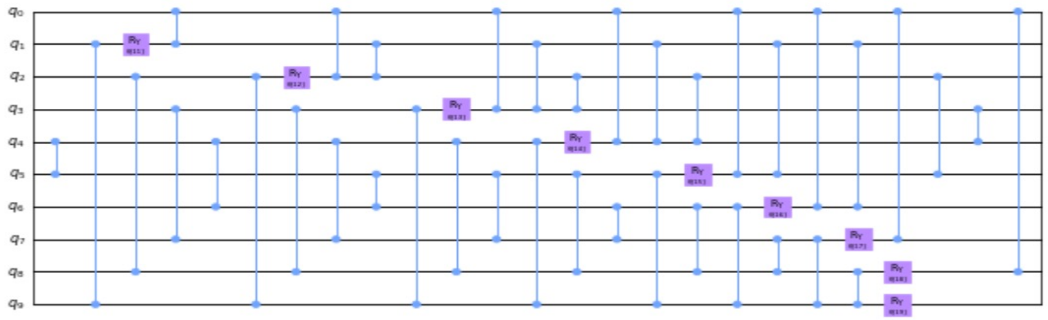
# Knapsack Problem with VQE

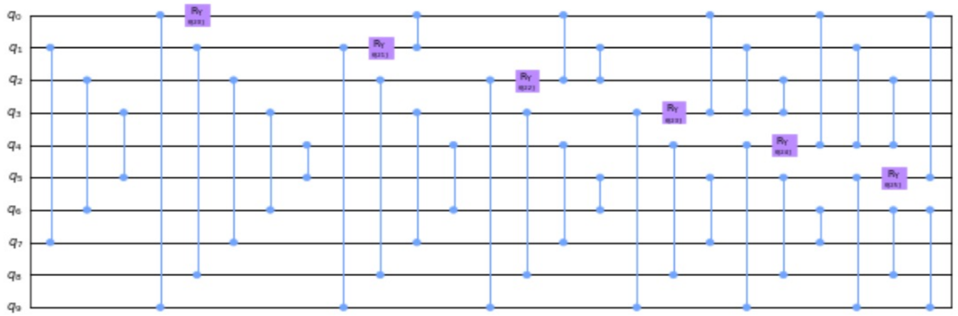
2. Make the VQE algorithm

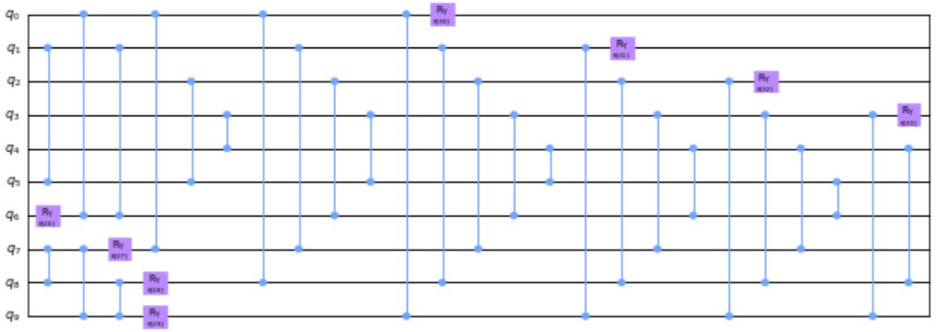
- Use SPSA optimizer, max_trials = 100

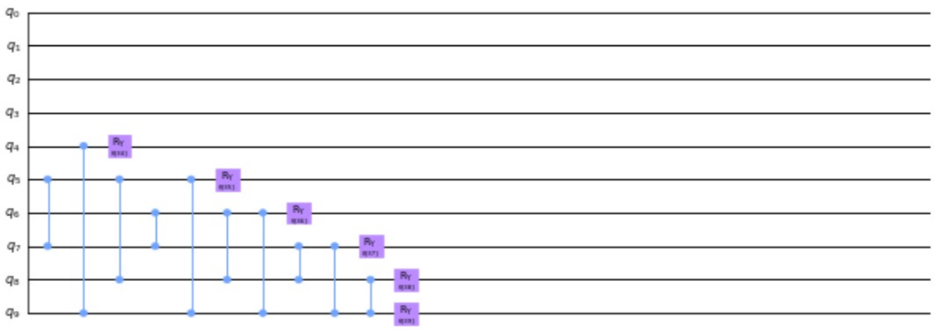
- Use TwoLocal (Ry rotation & cz entanglement)











# Knapsack Problem with VQE

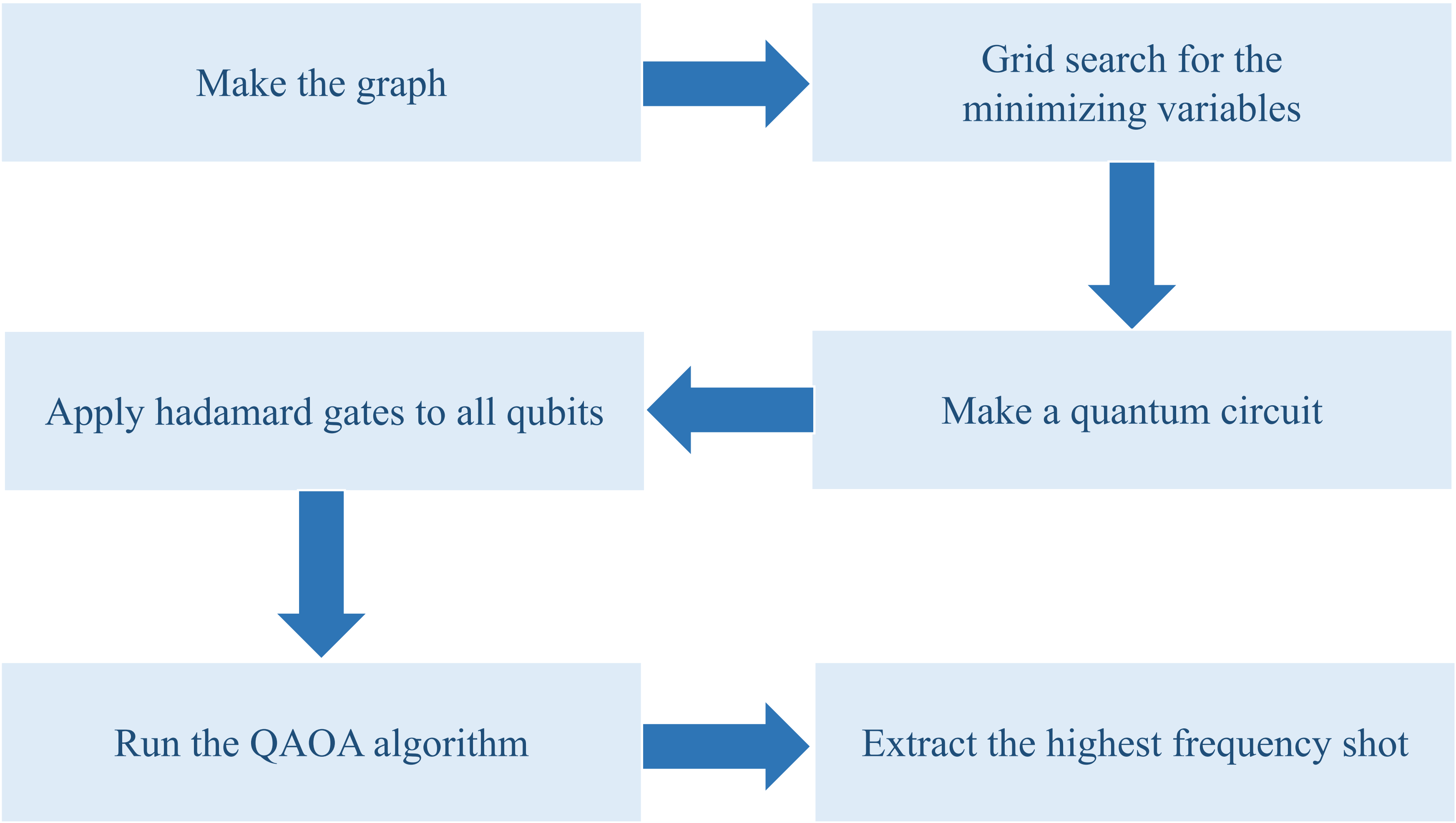
3. Run the VQE algorithm

- Use qasm simulator
- Choose the minimum eigenvector
- Extract the highest frequency shot

```
maxcount=0
for i in range(len(t)):
    a = t[i][1]
    if maxcount<a:
        maxcount=a

for i in range(len(t)):
    if t[i][1] == maxcount:
        shot=t[i][0]

print(shot[:len(values)])
```

```
1101
```

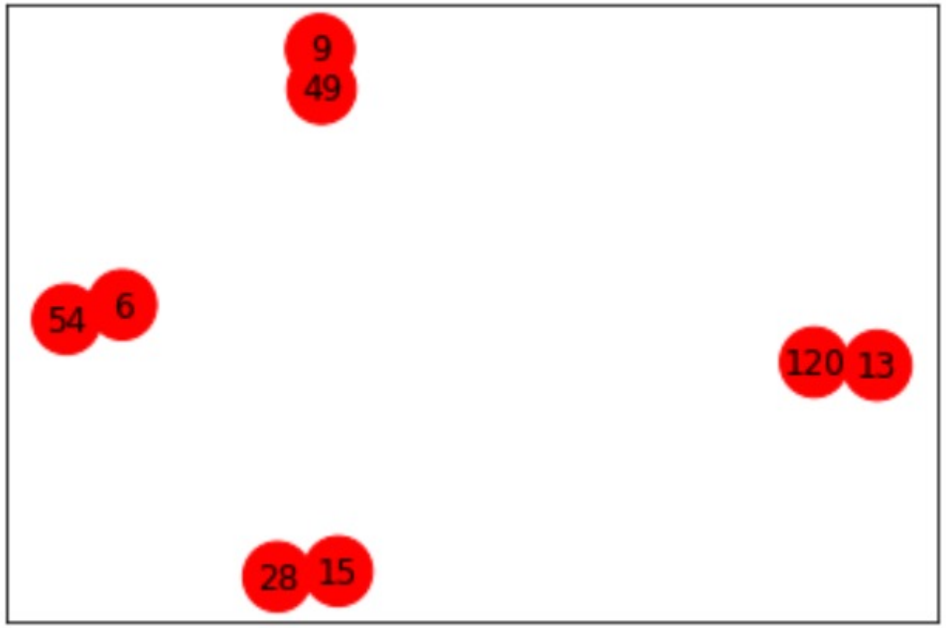
# Knapsack Problem with QAOA



# Knapsack Problem with QAOA

1. Make the graph

- Vertex : values
- Edge : values and weights set
- Add the vertex and edge in graph



# Knapsack Problem with QAOA

2. Grid search for the minimizing variables

- By qiskit QAOA guideline, the expectation value is :

$$F_1(\gamma,\beta) = 3 - \left( sin^2(2\beta)sin^2(2\gamma) - \frac{1}{2}sin(4\beta)sin(4\gamma) \right) \left(1 + cos^2(4\gamma)\right)$$

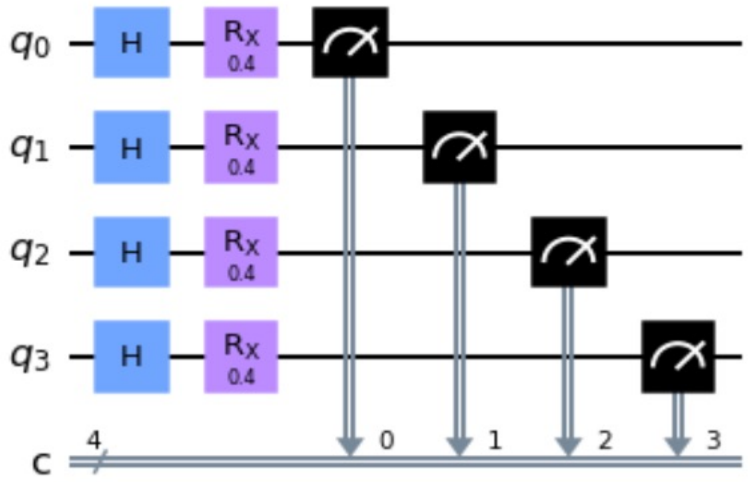
- Calculate F1 using a_gamma and a_beta

```
step_size   = 0.1;

a_gamma          = np.arange(0, np.pi, step_size)
a_beta           = np.arange(0, np.pi, step_size)
a_gamma, a_beta  = np.meshgrid(a_gamma,a_beta)

F1 = 3-(np.sin(2*a_beta)**2*np.sin(2*a_gamma)**2-0.5*np.sin(4*a_beta)
        *np.sin(4*a_gamma))*(1+np.cos(4*a_gamma)**2)

result = np.where(F1 == np.amax(F1))
a      = list(zip(result[0],result[1]))[0]

gamma  = a[0]*step_size;
beta   = a[1]*step_size;
```

# Knapsack Problem with QAOA

3. Make quantum circuit

- Apply Hadamard gates to all qubits
- Apply the Rx gates with angle beta to all qubits
- Measure the QAOA circuit




4. Run the QAOA algorithm

- Extract the highest frequency shot

```
import operator
circ={}
circ=QAOA_results.get_counts()

a=max(circ.items(), key=operator.itemgetter(1))[0]
b=max(circ.items(), key=operator.itemgetter(1))[1]
print(a)
print(b)
```

```
1101
76
```

# Conclusion

| | VQE | QAOA |
|---|---|---|
| Idea | minimum eigenvalue of H | grid search for the minimizing variables |
| Tools | pauli operator | graph |
| Qubits | (number of items + 1) * 2 | number of items |
| Gates | Ry | Rx |

- The two algorithms are the same in that they build and measure circuits and determine the most frequent value
- It spends a lot of time to run the algorithm with real quantum computer backend, so I just use qasm simulator