Improvement in Variational Quantum Algorithms by Measurement Simplification

2021.12.10

Jaehoon Hahm, Seoul National University Physics & Astronomy Department
Hayeon Kim, Seoul National University Electrical & Computer Engineering Department

Contents

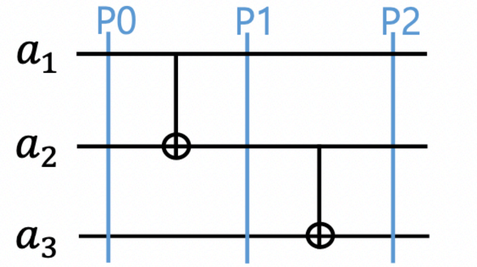
1. Simple rules in basic quantum circuits

CNOT 2개의 경우

6 Possible CNOT Gates

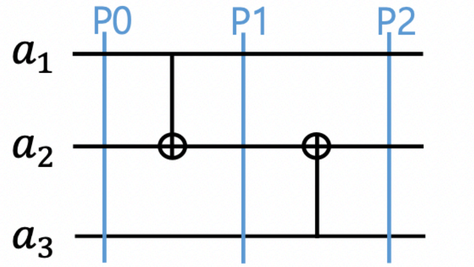
CNOT 이 2개가 있는 경우, simple rule 이 발견되는데, 이 구조는 VQC 구조와 유사한 점이 있다.

1.



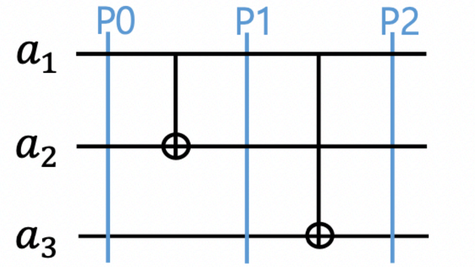
Propagation rule

2.

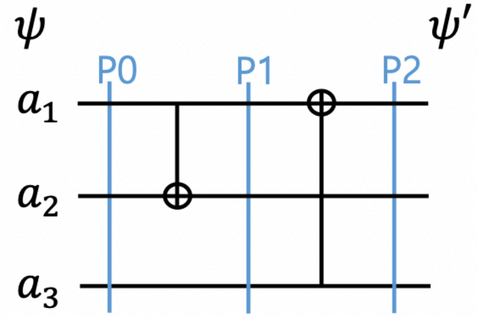


Adding rule(from independent source)

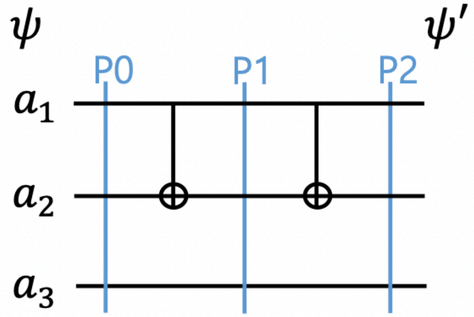
3.



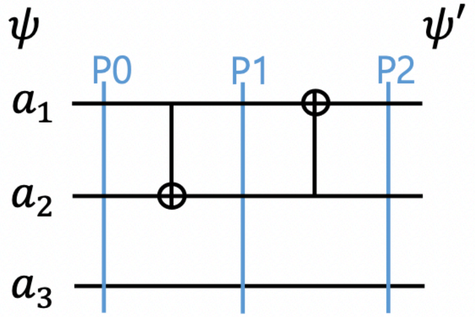
4.



5.

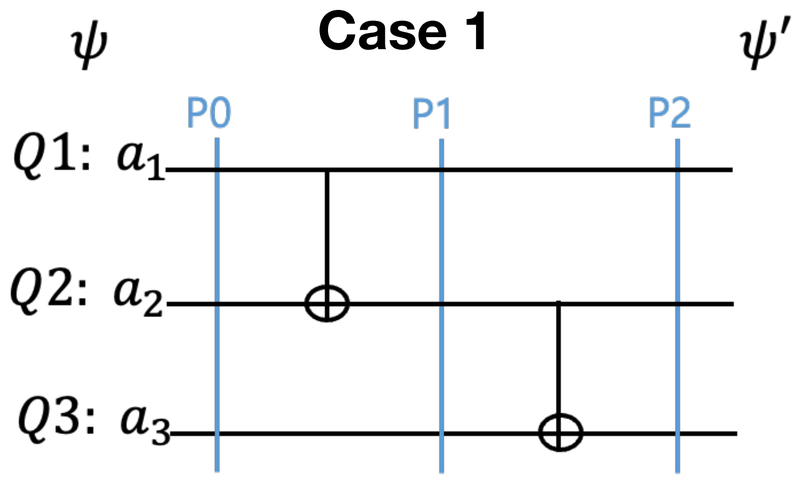


6.



Adding rule(dependent source))

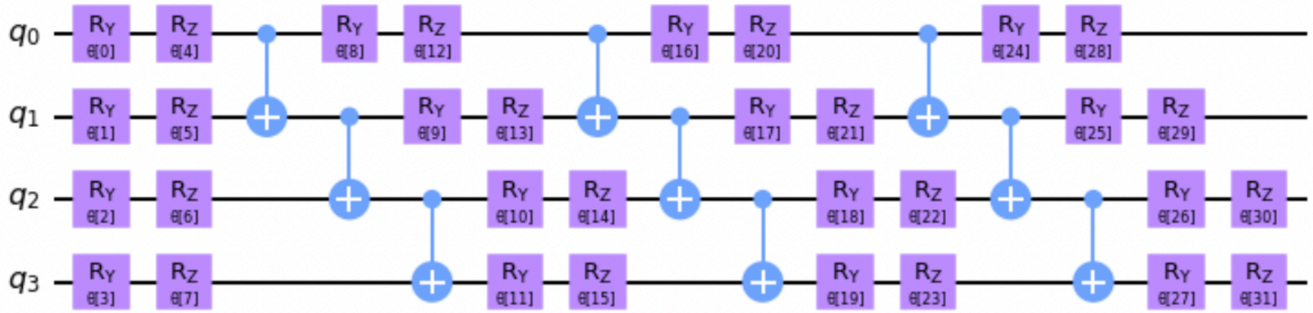
CNOT Rule



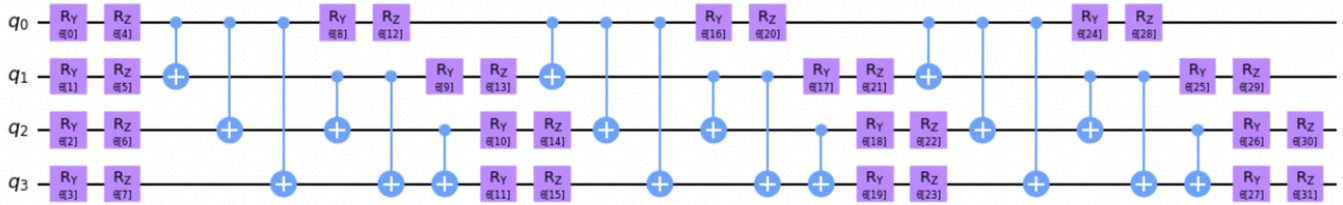
	Q1	Q2	Q3
P0	${a_1}^2$	${a_2}^2$	${a_3}^2$
P1	${a_1}^2$	${a_1}^2{a_2}^2 + {b_1}^2{b_2}^2$	${a_3}^2$
P2	${a_1}^2$	${a_1}^2{a_2}^2 + {b_1}^2{b_2}^2$	$({a_1}^2{a_2}^2 + {b_1}^2{b_2}^2){a_3}^2 + ({a_1}^2{b_2}^2 + {b_1}^2{a_2}^2){b_3}^2$

VQC 구조와의 유사성

==========Linear Entanglement:==========



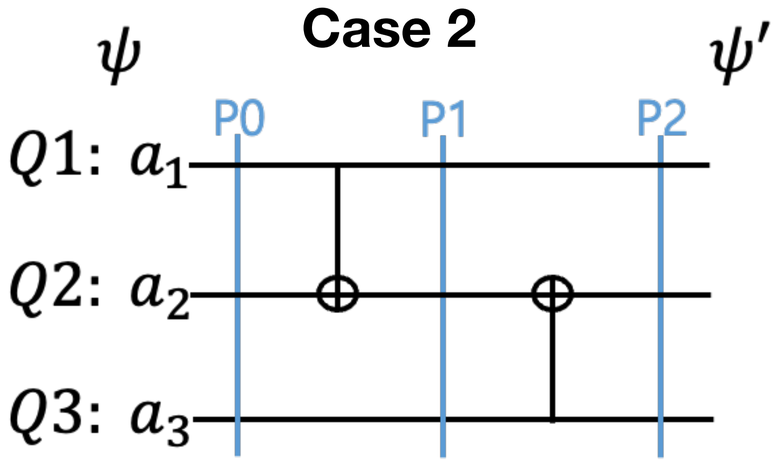
==========Full Entanglement:==========



왼쪽 회로는 H_2 분자의 바닥 상태 에너지를 Variational Method 로 계산하는 Variational Quantum Eigensolver (VQE) 의 회로이다. VQE 의 회로는 Variational Quantum Circuit, VQC의 한 종류로, 앞서 살펴본 CNOT Rule 에서 계산해본 일련의 CNOT 연결과 유사한 구조를 갖고 있다.

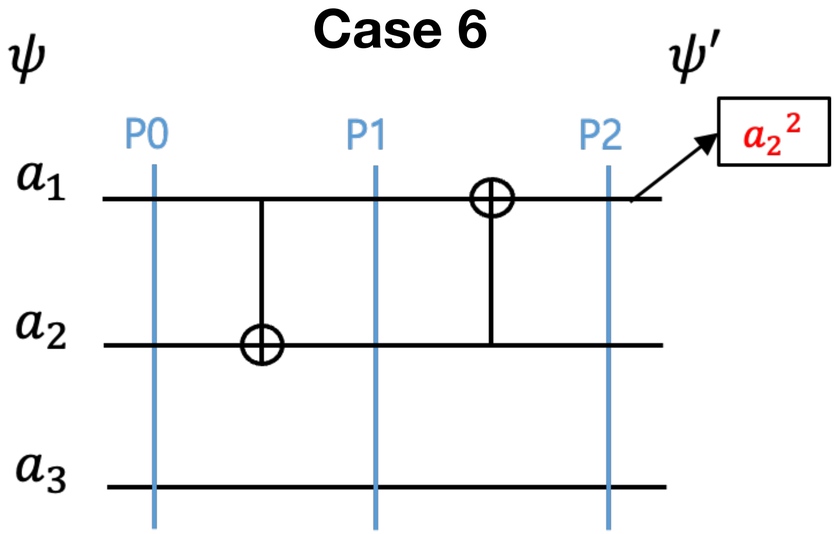
이를 계기로 다양한 종류의 VQC에서도 일련의 패턴을 찾아 expression simplification, 즉 speedup 이 가능할 것이란 구상을 할 수 있었다.

CNOT Rule



	Q1	Q2	Q3
P0	${a_1}^2$	${a_2}^2$	${a_3}^2$
P1	${a_1}^2$	${a_1}^2{a_2}^2 + {b_1}^2{b_2}^2$	${a_3}^2$
P2	${a_1}^2$	$({a_1}^2{a_2}^2 + {b_1}^2{b_2}^2){a_3}^2 + ({a_1}^2{b_2}^2 + {b_1}^2{a_2}^2){b_3}^2$	${a_3}^2$

CNOT Rule

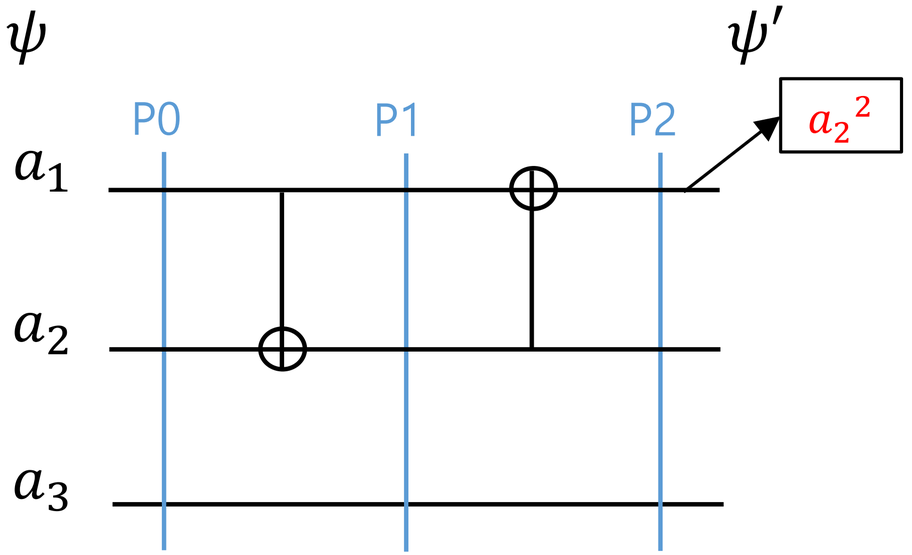


	Q1(Real)	Q1(Rule)	Q2	Q3
P0	a_1^2	a_1^2	a_2^2	a_3^2
P1	a_1^2	a_1^2	$a_1^2a_2^2 + b_1^2b_2^2$	a_3^2
P2	a_2^2	$a_2^2 - 2a_1^2b_1^2(a_2^2 - b_2^2)$	$a_1^2a_2^2 + b_1^2b_2^2$	a_3^2

Case 6: CNOT followed by CNOTed is the ‘injection’ of the CNOTed

CNOT 을 basis permutation operator 로 해석할 수 있음 :
(5 7)(6 8) (3 7)(4 8) =
(1 2 3 4 5 6 7 8)
(1 2 5 6 7 8 3 4)

3 - cycle, 즉 Case 6 는 3번 반복하면 identity 로 돌아오는 패턴을 갖고 있음.

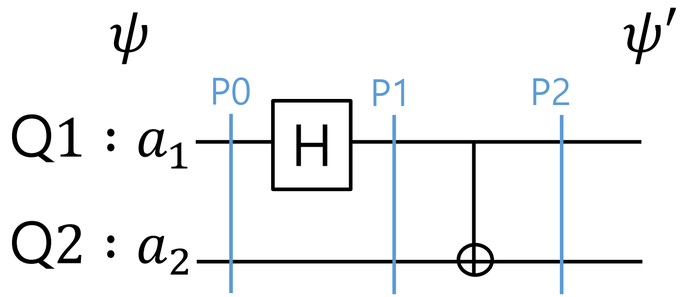


	Q1(Real)	Q1(Rule)	Q2	Q3
P0	$a_1{}^2$	$a_1{}^2$	$a_2{}^2$	$a_3{}^2$
P1	$a_1{}^2$	$a_1{}^2$	$a_1{}^2a_2{}^2 + b_1{}^2b_2{}^2$	$a_3{}^2$
P2	$a_2{}^2$	$a_2{}^2 - 2a_1{}^2b_1{}^2(a_2{}^2 - b_2{}^2)$	$a_1{}^2a_2{}^2 + b_1{}^2b_2{}^2$	$a_3{}^2$

H gate 를 추가한 후 Rule

H 가 추가되는 순간, entanglement 를 고려해줘야 하고, 다음 논의는 separable input state 을 가정하고 한 것이다.

H rule 1: Non-recurrent



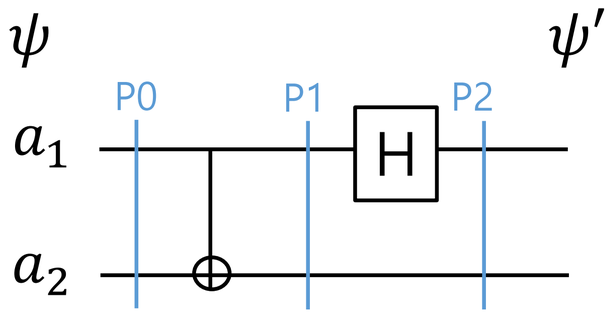
	Q1	Q2
P0	${a_1}^2$	${a_2}^2$
P1	$\frac{1}{2}+a_1b_1$	${a_2}^2$
P2	$\frac{1}{2}+a_1b_1$	$\frac{1}{2}+a_1b_1({a_2}^2-{b_2}^2)$

- H gate is non recurrent: Q1은 H 후 CNOT gate 해주어도 변하지 않음

H gate 를 추가한 후 Rule

H 가 추가되는 순간, entanglement 를 고려해줘야 하고, 다음 논의는 separable input state 을 가정하고 한 것이다.

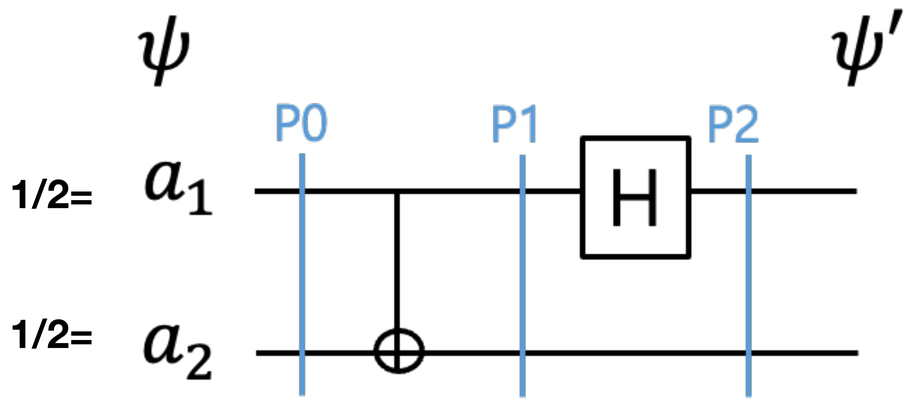
H rule 2: recurrent



	Q1	Q2
P0	${a_1}^2$	${a_2}^2$
P1	${a_1}^2$	${a_1}^2{a_2}^2 + {b_1}^2{b_2}^2$
P2	$\frac{1}{2} + a_1 b_1 (2a_2 b_2)$	${a_1}^2{a_2}^2 + {b_1}^2{b_2}^2$

➢ H gate is recurrent: Q1은 H gate를 적용할 때, 그 전 CNOT gate를 기억함

H rule 2 에 대한 분석



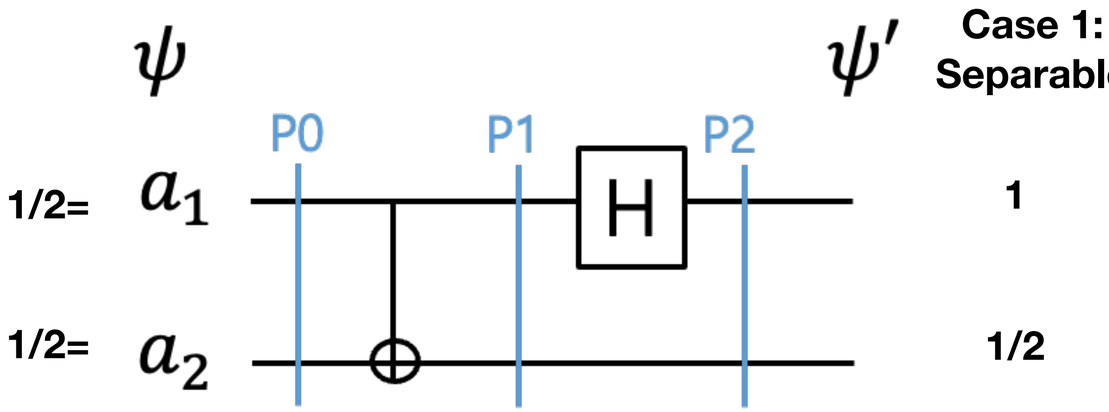
	Case 1: Separable	Case 2: Entangled
	1	1
	1/2	1

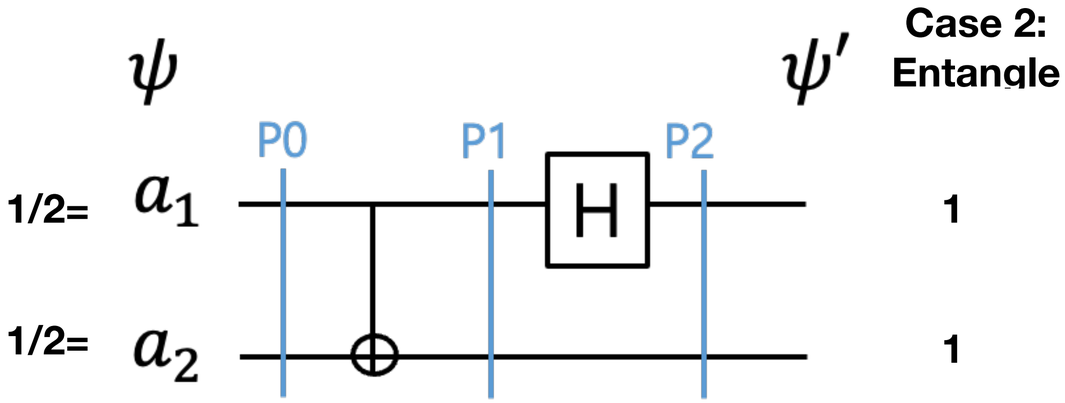
Case 1 (Separable) : $\frac{1}{2}(|0\rangle + |1\rangle)(|0\rangle + |1\rangle)$

Case 2 (Entangled) : $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$

H rule 2 를 분석해보면, input state 가 separable 인지 entangled 인지에 따라 measurement probability 가 달라지는 것을 확인할 수 있다.

즉, input state 의 entanglement 에 대한 정보를 고려해줘야 한다.





Case 1 (Separable) :

$$\frac{1}{2}(|0\rangle+|1\rangle)(|0\rangle+|1\rangle)$$

$$=\frac{1}{2}(|00\rangle+|01\rangle+|10\rangle+|11\rangle)$$

$$\xrightarrow{CNOT_{12}}\frac{1}{2}(|00\rangle+|01\rangle+|10\rangle+|11\rangle)$$ (**no affect of CNOT**)

$$\xrightarrow{H_1}\frac{1}{\sqrt{2}}|0\rangle(|0\rangle+|1\rangle)$$

Case 2 (Entangled) :

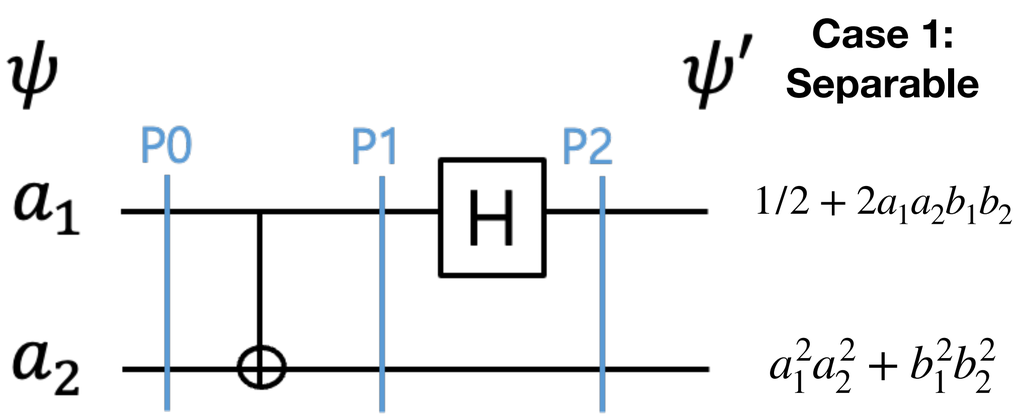
$$\frac{1}{2}(|00\rangle+|11\rangle)$$

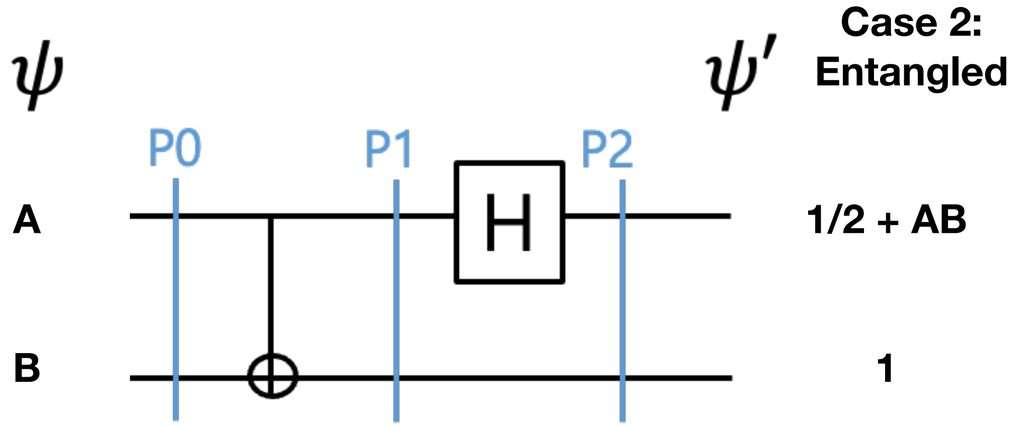
$$\xrightarrow{CNOT_{12}}\frac{1}{\sqrt{2}}(|00\rangle+|10\rangle)$$

$$=|+\rangle|0\rangle$$

$$\xrightarrow{H_1}|00\rangle$$

Output of circuit depends on the entanglement of input.





Case 1 (Separable) :

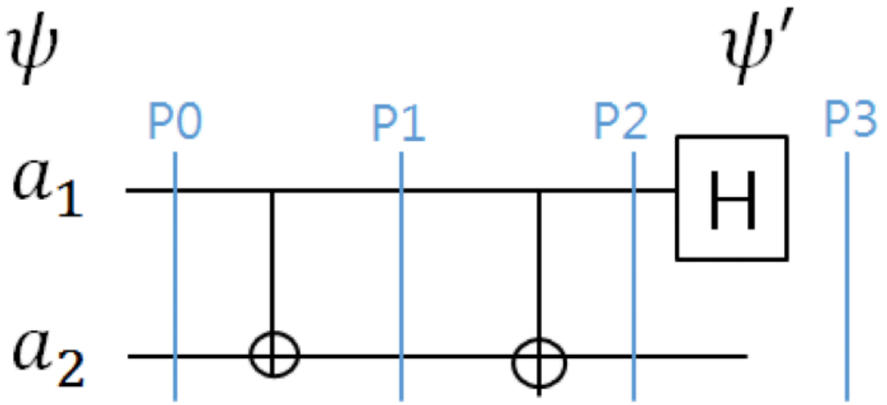
$$(a_1|0\rangle + b_1|1\rangle)(a_2|0\rangle + b_2|1\rangle)$$
$$= a_1a_2|00\rangle + a_1b_2|01\rangle + b_1a_2|10\rangle + b_1b_2|11\rangle$$
$$\xrightarrow{CNOT_{12}} a_1a_2|00\rangle + a_1b_2|01\rangle + b_1b_2|10\rangle + b_1a_2|11\rangle$$
$$\xrightarrow{H_1} a_1\frac{|0\rangle + |1\rangle}{\sqrt{2}}(a_2|0\rangle + b_2|1\rangle) + b_1\frac{|0\rangle - |1\rangle}{\sqrt{2}}(b_2|0\rangle + a_2|1\rangle)$$
$$= \frac{1}{\sqrt{2}}[(a_1a_2 + b_1b_2)|00\rangle + (a_1b_2 + b_1a_2)|01\rangle$$
$$+(a_1a_2 - b_1b_2)|10\rangle + (a_1b_2 - b_1a_2)|11\rangle]$$

Case 2 (Entangled) : $(|A|^2 + |B|^2 = 1)$

$$A|00\rangle + B|11\rangle$$
$$\xrightarrow{CNOT_{12}} A|00\rangle + |10\rangle)$$
$$\xrightarrow{H_1} \frac{1}{\sqrt{2}}((A + B)|0\rangle + (A - B)|1\rangle)|0\rangle$$

Output of circuit depends on the entanglement of input.

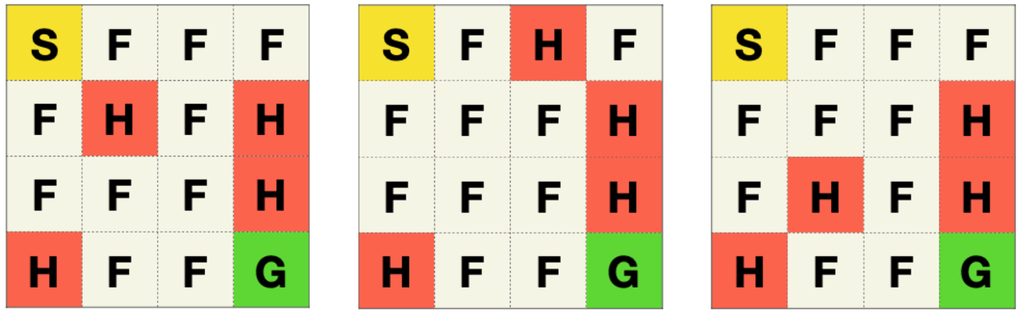
H gate rule 3



	Q1	Q2
P0	${a_1}^2$	${a_2}^2$
P1	a_1^2	${a_1}^2{a_2}^2 + {b_1}^2{b_2}^2$
P2	a_1^2	${a_2}^2$
P3	$\frac{1}{2}+a_1b_1$	${a_2}^2$

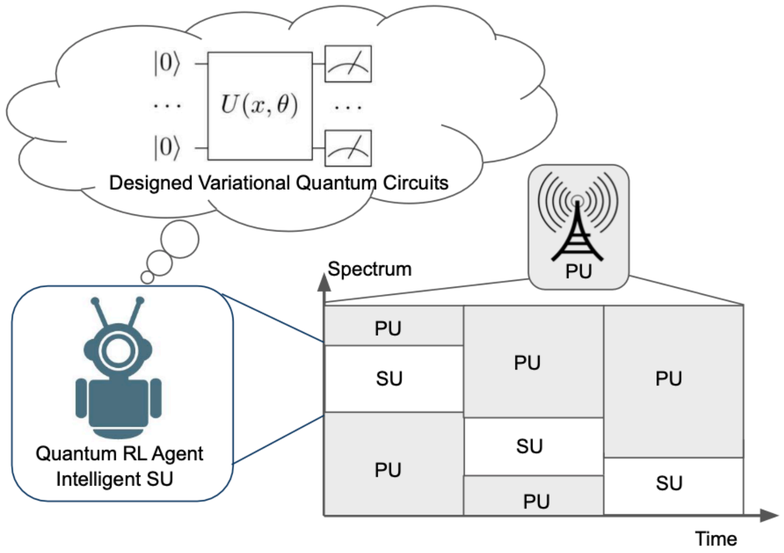
2. Application in Quantum Deep Reinforcement Learning(QDRL)

VQC for Deep Reinforcement Learning



(a) Frozen-Lake

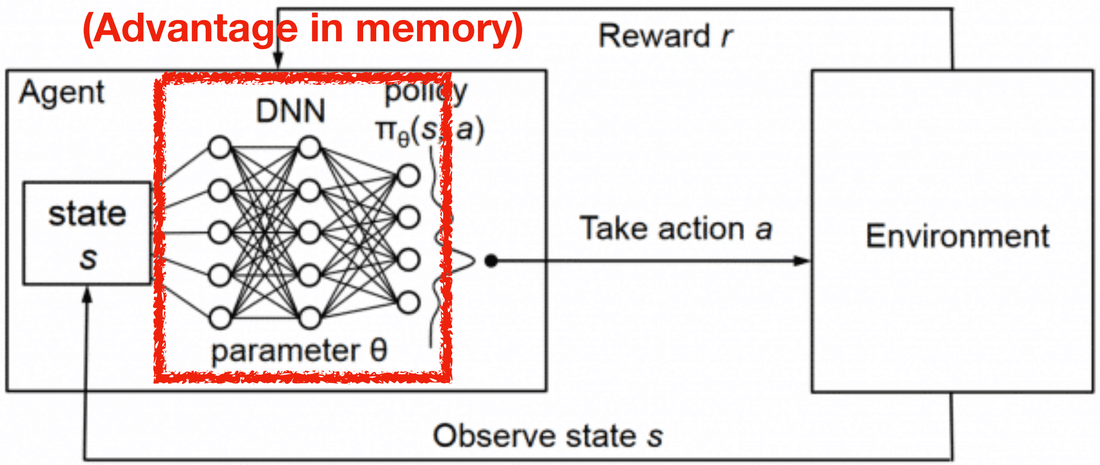
Location	Reward
HOLE	-0.2
GOAL	+1.0
OTHER	-0.01

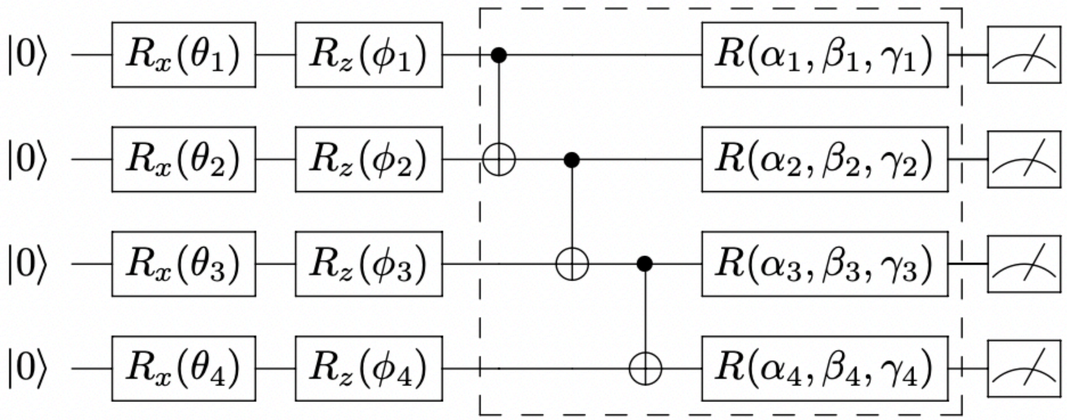


S. Y. -C. Chen, C. -H. H. Yang, J. Qi, P. -Y. Chen, X. Ma and H. -S. Goan, "Variational Quantum Circuits for Deep Reinforcement Learning," in IEEE Access, vol. 8, pp. 141007-141024, 2020, doi: 10.1109/ACCESS.2020.3010470.

VQC for Deep Reinforcement Learning

Make this part in Quantum Circuit!
(Advantage in memory)





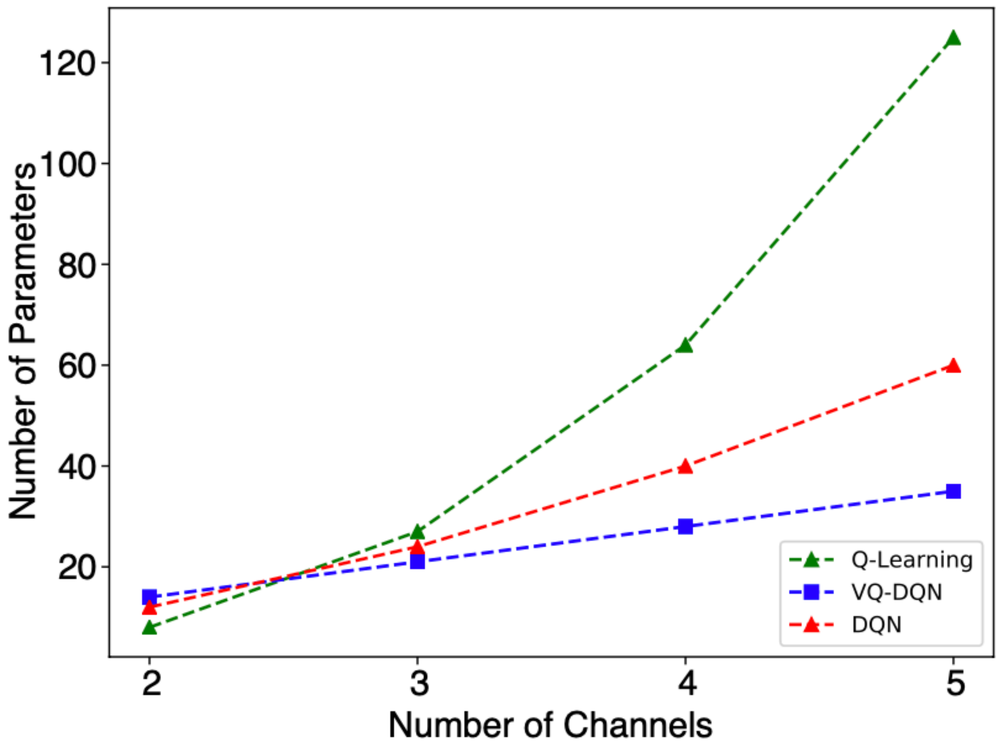
- 이에 따라, Variational Quantum Circuit (VQC) 에서의 활용을 제안한다.

- 왼쪽 회로는 VQC 의 대표적인 경우로, expressibility 가 높다고 알려져 있어 Quantum Chemistry, Quantum Machine Learning 에서 활용되고 있다.

Reference: Chen et.al.,“Variational Quantum Circuits for Deep Reinforcement Learning”

Memory Advantage

F. QUANTUM ADVANTAGE ON MEMORY CONSUMPTION



Quantum Deep Q-Learning 을 이용하면 필요한 메모리 용량이 기하급수적으로 작아진다.

	Classical Q-learning	Deep Q-learning	Quantum DQL
# of parameters	$O(n^3)$	$O(n^2)$	$O(n)$

Variational Quantum Circuits

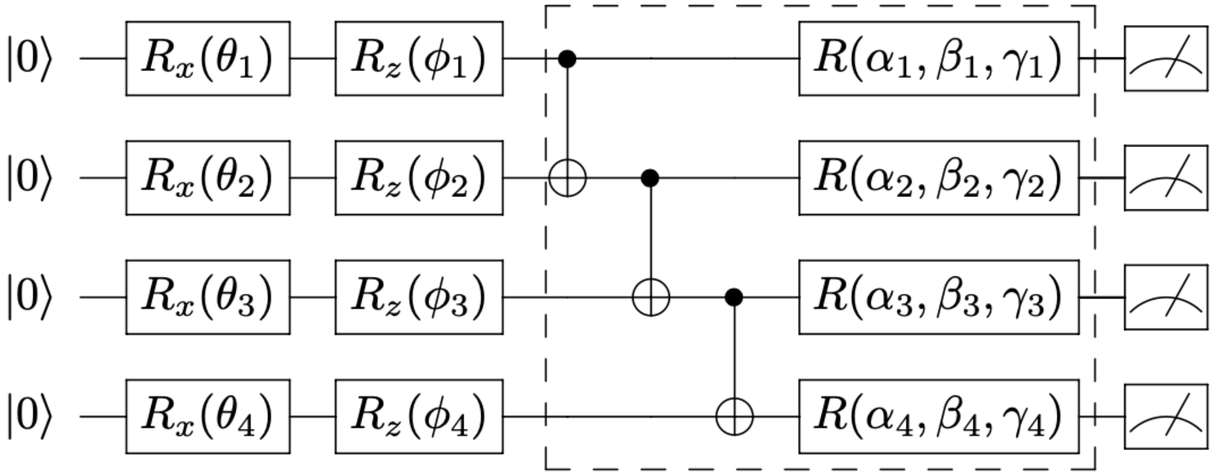


FIGURE 5: **Generic variational quantum circuit architecture for the deep Q network (VQ-DQN).** The single-

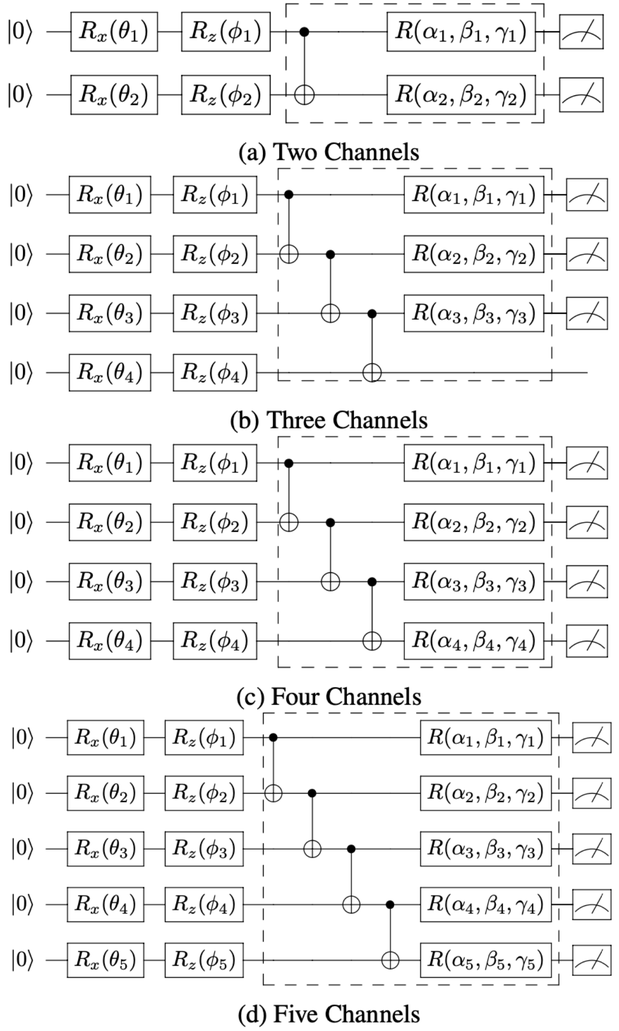
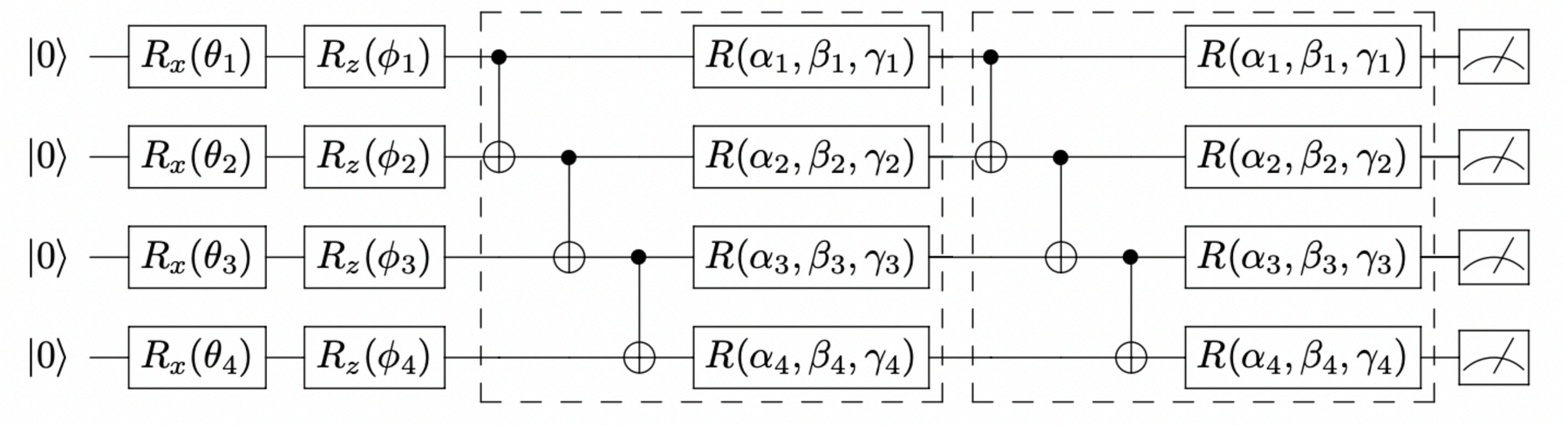
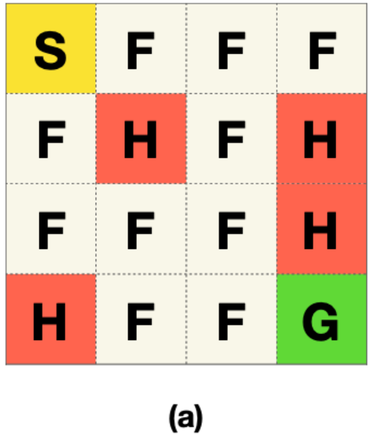


FIGURE 6: **Variational quantum circuits for the cognitive-radio experiments.** The basic architecture is

강화학습에서 VQC 의 활용





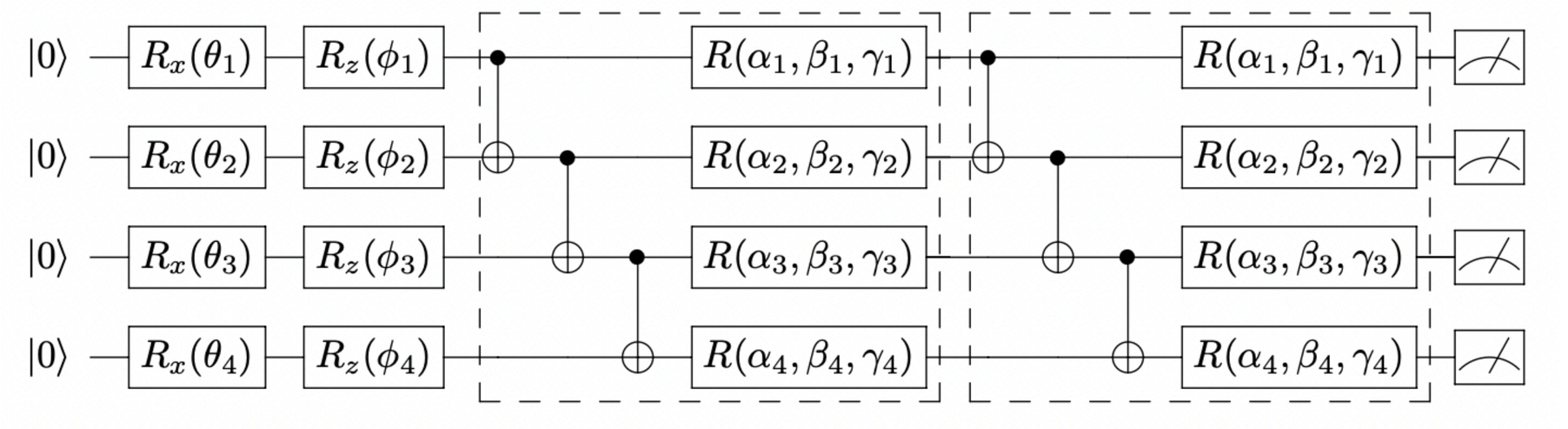
(a)

Initialization of “state” **Iterative optimization** **Measuring “action”**

미로찾기와 같은 강화학습 문제에서 위와 같은 VQC 를 이용할 수 있다.

고전적인 방법의 Deep Q-Network (DQN) 보다 VQC 를 활용한 DQN 이 훨씬 더 적은 parameter 의 개수로 학습시킬 수 있고, 그에 따라 메모리 용량을 크게 줄일 수 있다. (Number of parameters for Classical DQN : O(n^3), VQC-DQN: O(n))

강화학습에서의 VQC 활용

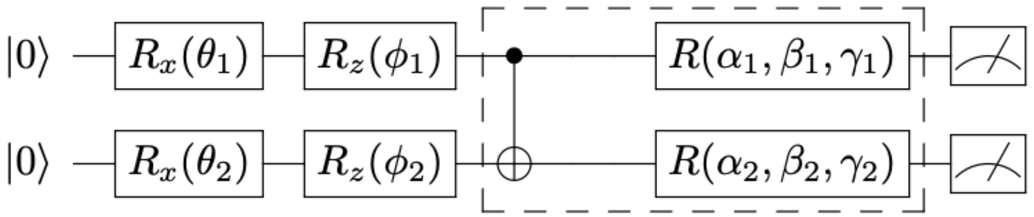


Initialization of “state” **Iterative optimization** **Measuring “action”**

만약 위와 같은 VQC의 응용 사례가 각각의 큐빗들이 0으로 측정될 확률만을 필요로 하는 과정이라면, 전체 state vector 를 계산하지 않고, 마지막 measurement expression 만이 필요하다. 이 expression 은 앞서 살펴보았듯이 simplify 가 가능한 경우가 있고, 여기서 speedup 을 이끌어낼 수 있다.

이러한 양자회로의 사용은 Deep Q-Learning 과 같은 신경망 학습과 강화학습과 같은 상황에서 훨씬 적은 메모리로 같은 퍼포먼스를 내는 이점이 있다.

Measurement Simplification



(a) Two Channels

- 왼쪽과 같은 2 qubit VQC 의 결과를 계산해보았다.
- 왼쪽의 경우에 output은 1st qubit, 2nd qubit 이 0으로 측정될 확률이다.
- 연구의 목표는 마지막에 output 에서는 statevector의 진폭들이 제곱되고 더해진 measurement probability 만이 중요하기 때문에 , 이 expression 을 simplify 해서 얻어지는 computational benefit 을 분석해 보는 것이다.
- 가정: input state 의 진폭 정보가 실수라 가정함. 즉, input = (cos(theta), sin (theta)) 형태. (이 가정은 추후 완화 가능)

```
%result is the output statevector
result = rotation * CNOT * init;

%r is the Prob(0) for the 1st qubit
r = result(1)*conj(result(1)) + result(2)*conj(result(2))

simplify(r)
simplify(r,10)
simplify(r,30)
simplify(r,50)
simplify(r,100)
simplify(r,200)
```

MATLAB 에서의 구현

Without simplify… expression is very long and complicated…

$$\left(\cos\left(\frac{\beta_1}{2}\right)\cos\left(\frac{\beta_2}{2}\right)\sigma_4\sigma_5\sigma_7\sigma_8\cos(x_1)\cos(x_2)-\cos\left(\frac{\beta_1}{2}\right)\sin\left(\frac{\beta_2}{2}\right)\sigma_4\sigma_5\sigma_7\sigma_8\cos(x_1)\sin(x_2)-\cos\left(\frac{\beta_1}{2}\right)\sin\left(\frac{\beta_2}{2}\right)\sigma_4\sigma_5\sigma_6\sigma_7\sin(x_1)\sin(x_2)+\sin\left(\frac{\beta_1}{2}\right)\sin\left(\frac{\beta_2}{2}\right)\sigma_4\sigma_5\sigma_6\sigma_7\cos(x_2)\sin(x_1)\right)\left(\cos\left(\frac{\beta_1}{2}\right)\cos\left(\frac{\beta_2}{2}\right)\sigma_6\sigma_7\sigma_8\sigma_9\cos(x_1)\cos(x_2)-\cos\left(\frac{\beta_1}{2}\right)\sin\left(\frac{\beta_2}{2}\right)\sigma_6\sigma_7\sigma_8\sigma_9\cos(x_1)\sin(x_2)-\cos\left(\frac{\beta_1}{2}\right)\sin\left(\frac{\beta_2}{2}\right)\sigma_6\sigma_7\sigma_8\sigma_9\sin(x_1)\sin(x_2)+\sin\left(\frac{\beta_1}{2}\right)\sin\left(\frac{\beta_2}{2}\right)\sigma_6\sigma_7\sigma_8\sigma_9\cos(x_2)\sin(x_1)\right)+\left(\cos\left(\frac{\beta_1}{2}\right)\cos\left(\frac{\beta_2}{2}\right)\sigma_4\sigma_5\sigma_7\sigma_8\cos(x_1)\sin(x_2)+\cos\left(\frac{\beta_1}{2}\right)\sin\left(\frac{\beta_2}{2}\right)\sigma_4\sigma_5\sigma_7\sigma_8\cos(x_1)\cos(x_2)-\cos\left(\frac{\beta_1}{2}\right)\sin\left(\frac{\beta_2}{2}\right)\sigma_4\sigma_5\sigma_6\sigma_8\cos(x_2)\sin(x_1)-\sin\left(\frac{\beta_1}{2}\right)\sin\left(\frac{\beta_2}{2}\right)\sigma_4\sigma_5\sigma_6\sigma_8\sin(x_1)\sin(x_2)\right)\left(\cos\left(\frac{\beta_1}{2}\right)\cos\left(\frac{\beta_2}{2}\right)\sigma_6\sigma_7\sigma_8\sigma_9\cos(x_1)\sin(x_2)+\cos\left(\frac{\beta_1}{2}\right)\sin\left(\frac{\beta_2}{2}\right)\sigma_6\sigma_7\sigma_8\sigma_9\cos(x_1)\cos(x_2)-\cos\left(\frac{\beta_1}{2}\right)\sin\left(\frac{\beta_2}{2}\right)\sigma_6\sigma_7\sigma_8\sigma_9\cos(x_2)\sin(x_1)-\sin\left(\frac{\beta_1}{2}\right)\sin\left(\frac{\beta_2}{2}\right)\sigma_6\sigma_7\sigma_8\sigma_9\sin(x_1)\sin(x_2)\right)$$

where

$\sigma_1 = e^{-\frac{\gamma_1 i}{2}}$
$\sigma_2 = e^{-\frac{\gamma_2 i}{2}}$
$\sigma_3 = e^{-\frac{\gamma_3 i}{2}}$
$\sigma_4 = e^{-\frac{\gamma_4 i}{2}}$
$\sigma_5 = e^{\frac{\gamma_4 i}{2}}$
$\sigma_6 = e^{\frac{\gamma_3 i}{2}}$
$\sigma_7 = e^{\frac{\gamma_2 i}{2}}$
$\sigma_8 = e^{\frac{\gamma_1 i}{2}}$

With simplify(r, steps), # of terms decreases dramatically

Default

$$2\sigma_1\cos(x_1)^2-\cos(x_1)^2-\sigma_1-2\cos\left(\frac{\beta_1}{2}\right)\sin\left(\frac{\beta_1}{2}\right)e^{-\gamma_1 i}\cos(x_1)\cos(x_2)\sin(x_1)\sin(x_2)-2\cos\left(\frac{\beta_1}{2}\right)\sin\left(\frac{\beta_1}{2}\right)e^{\gamma_1 i}\cos(x_1)\cos(x_2)\sin(x_1)\sin(x_2)+1$$

where

$$\sigma_1=\cos\left(\frac{\beta_1}{2}\right)^2$$

steps: 30

$$2\cos\left(\frac{\beta_1}{2}\right)^2\cos(x_1)^2-\cos\left(\frac{\beta_1}{2}\right)^2-4\sin\left(\frac{\beta_1}{2}\right)\cos(\gamma_1)\cos(x_2)\sin(x_1)\sin(x_2)\cos\left(\frac{\beta_1}{2}\right)\cos(x_1)-\cos(x_1)^2+1$$

steps: 100

$$\cos(\beta_1)\cos(x_1)^2-4\cos\left(\frac{\beta_1}{2}\right)\sin\left(\frac{\beta_1}{2}\right)\cos(\gamma_1)\cos(x_2)\sin(x_1)\sin(x_2)\cos(x_1)-\frac{\cos(\beta_1)}{2}+\frac{1}{2}$$

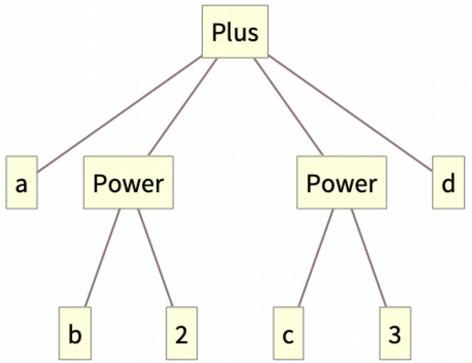
steps: 200

$$\frac{\left(2\cos\left(\frac{\beta_1}{2}\right)^2-1\right)\left(2\cos(x_1)^2-1\right)}{2}-4\cos\left(\frac{\beta_1}{2}\right)\sin\left(\frac{\beta_1}{2}\right)\cos(\gamma_1)\cos(x_1)\cos(x_2)\sin(x_1)\sin(x_2)+\frac{1}{2}$$

Simplification 후에 항이 큰 차이로 줄어드는 것을 확인하였다.
Computational benefit 이 얼마인지 정량화할 수 있는 metric 을 찾은 후, 3장에서 다양한 VQC 에서의 benefit 이 얼마인지 비교하였다.

Leafcount

```
In[1]:= TreeForm[a + b^2 + c^3 + d]
```



LeafCount [*expr*]
gives the total number of indivisible subexpressions in *expr*.

- LeafCount gives a measure of the total "size" of an expression.
- LeafCount counts the number of subexpressions in *expr* that correspond to "leaves" on the expression tree.
- LeafCount is based on FullForm representation of expressions.
- Numbers with heads Rational and Complex are treated as composite objects, just as in FullForm.

Leafcount 는 expression 의 total size 를 정의하는 mathematica 의 함수이고, 이를 이용해서 하나의 expression 의 “크기” 를 측정했다.
Expression 의 computation time이 Leafcount 에 비례할 것이라 가정하고 분석을 진행했다.

In[12]:=
```
n = 4;
Rx[θ_] := MatrixExp[-i * θ * PauliMatrix[1] / 2]
Ry[θ_] := MatrixExp[-i * θ * PauliMatrix[2] / 2]
Rz[θ_] := MatrixExp[-i * θ * PauliMatrix[3] / 2]
Id = IdentityMatrix[2];
H = HadamardMatrix[2];
R[a_, b_, c_] := Rz (a).Ry (b).Rz (c)

CNOT[n_, i_, j_] := (
mat = IdentityMatrix[2^n]; ind = 2^(n - 1 - i) + 2^(n - 1 - j);
mat[[ind, ind]] = -1;
mat
);

init0 = {{1}, {0}};
init = init0;
For[i = 0, i < n - 1, i++, init = KroneckerProduct[init, init0]]
```

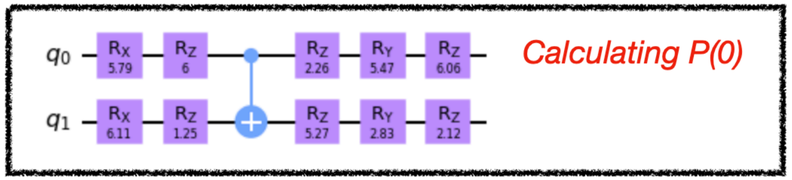
In[33]:=
```
sv = KroneckerProduct[Rx[a1].Rz[b1], Rx[a2].Rz[b2], Rx[a3].Rz[b3], Rx[a4].Rz[b4]] .init;
sv = CNOT[n, 3, 2].sv;
sv = CNOT[n, 2, 1].sv;
sv = CNOT[n, 1, 0].sv;

result = 0;
For[i = 1, i ≤ 2^(n - 1), i++, result = result + Abs[sv[[i, 1]]^2];
result;
LeafCount[result]
result = Simplify[result, a1 ∈ Reals && a2 ∈ Reals && a3 ∈ Reals && a4 ∈ Reals &&
    b1 ∈ Reals && b2 ∈ Reals && b3 ∈ Reals && b4 ∈ Reals];
LeafCount[result]
```

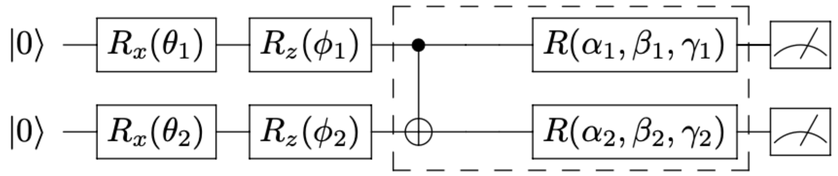
Out[40]= 321

Out[42]= 8

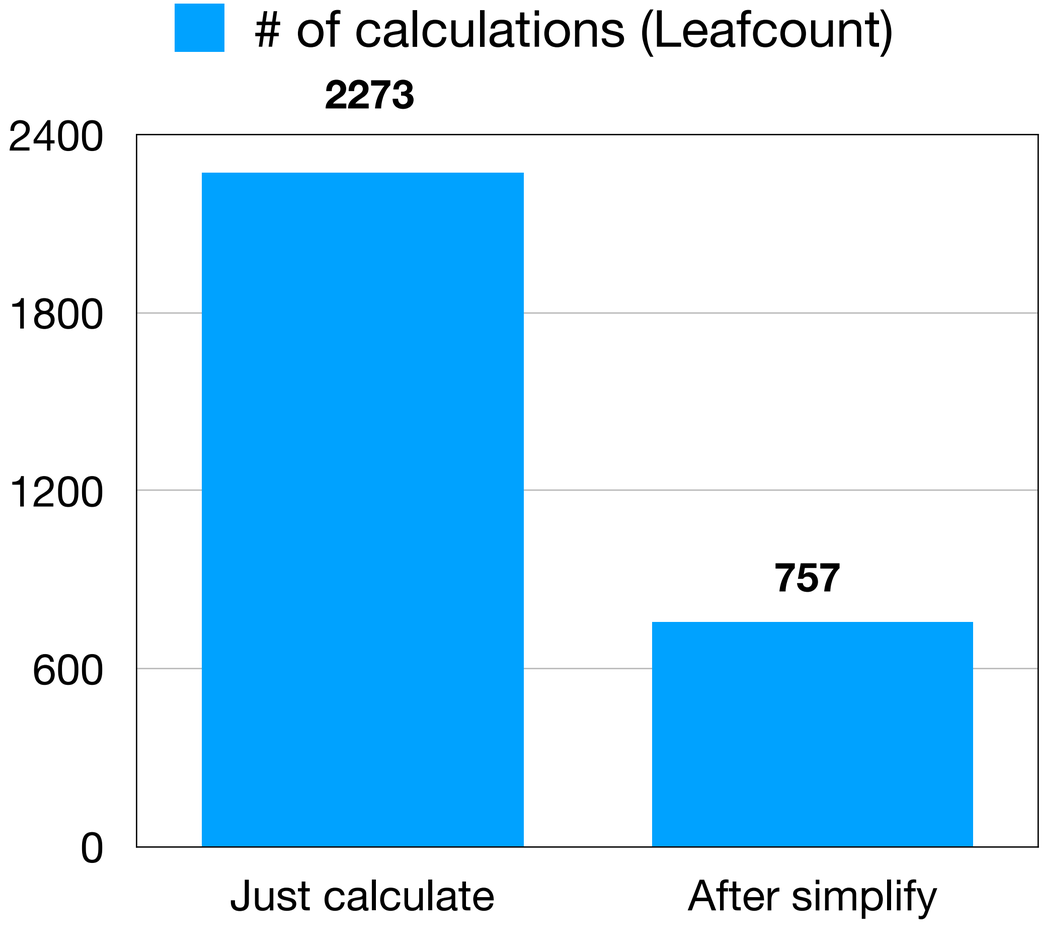
N=2, # of gates = 10, # of layers = 1

VQC circuit used in Quantum Reinforcement Learning



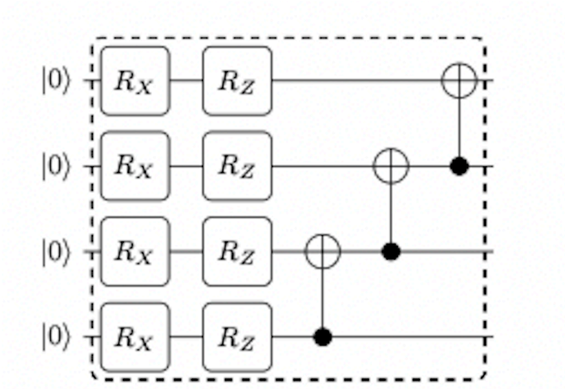


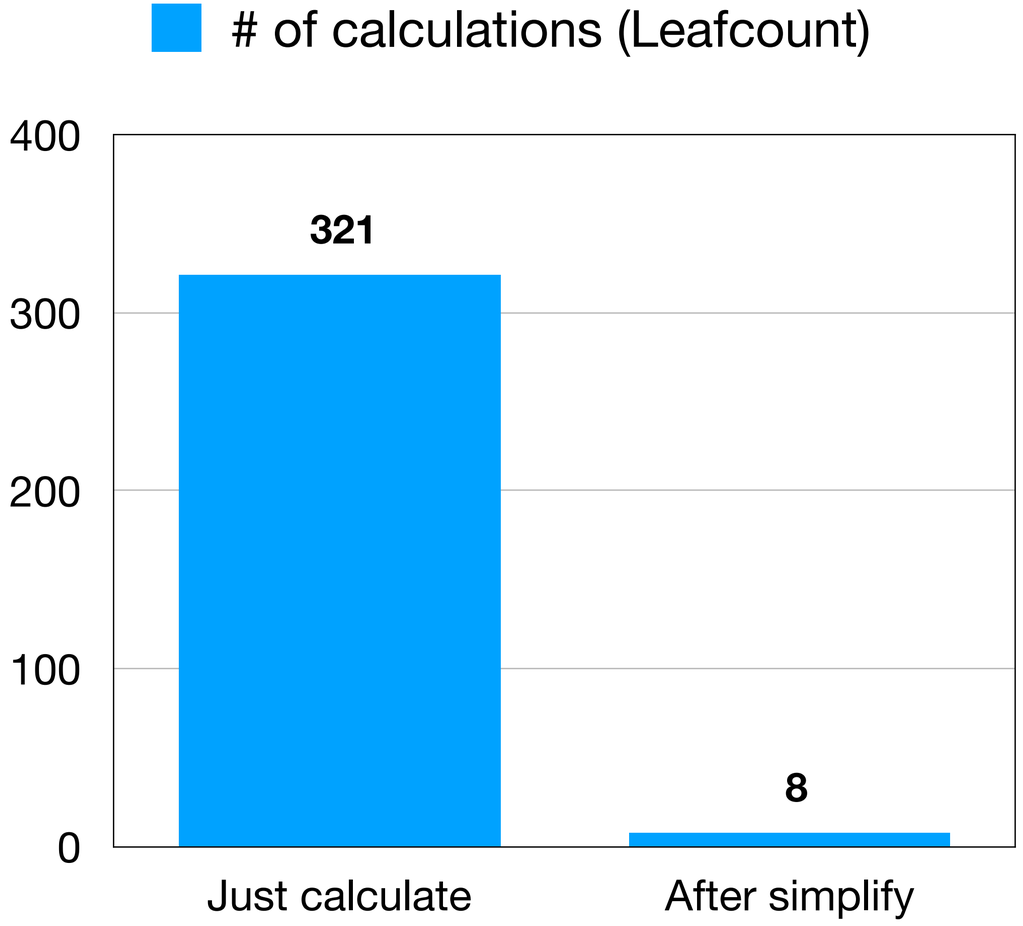
(a) Two Channels



N=4, # of gates = 8, # of layers = 1

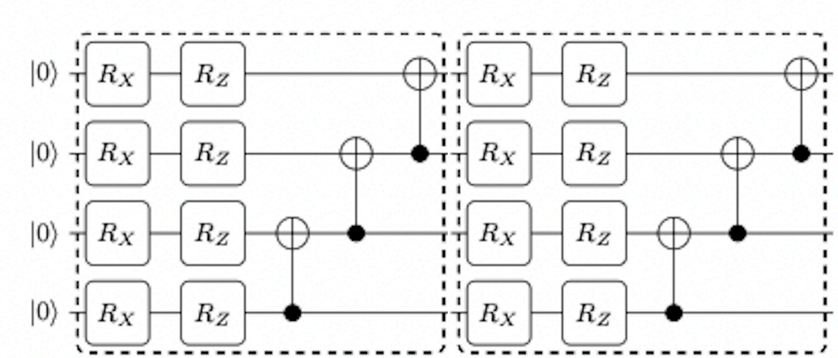
Various VQCs …

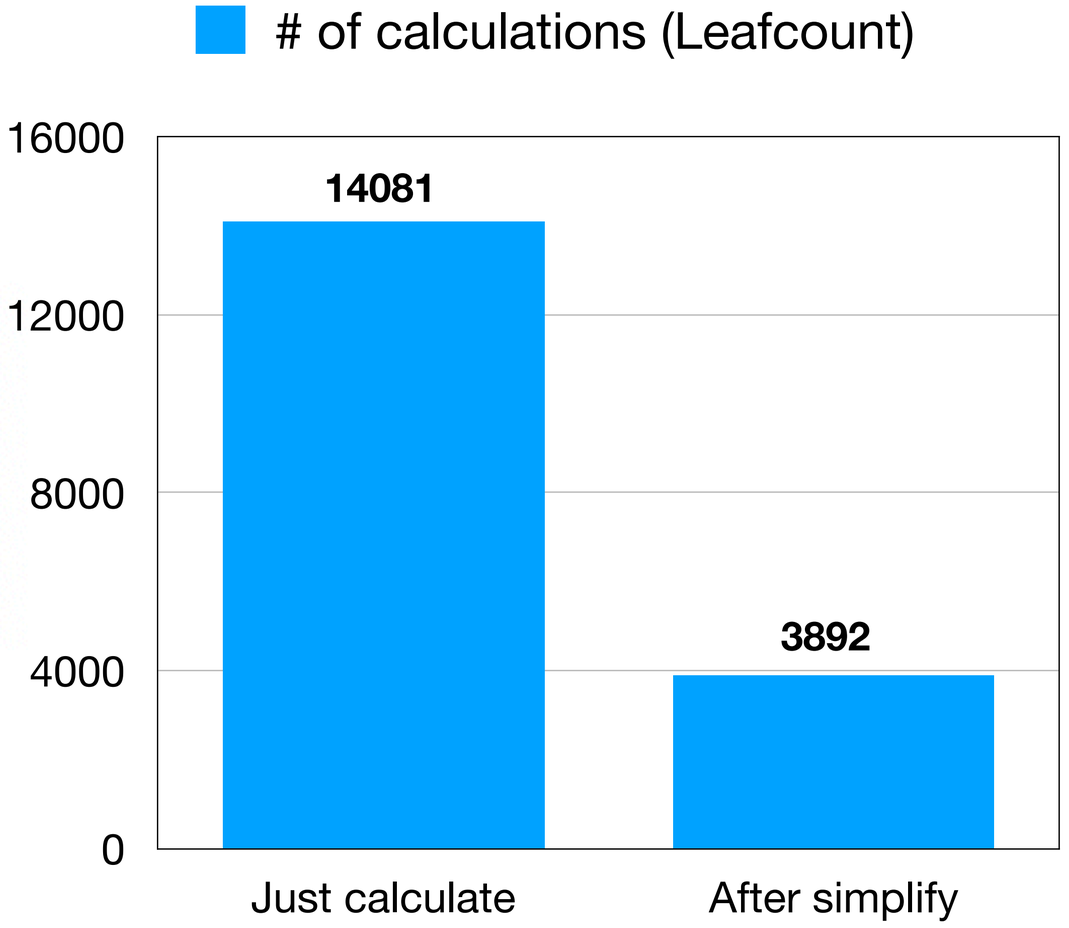




N=4, # of gates = 16, # of layers = 2

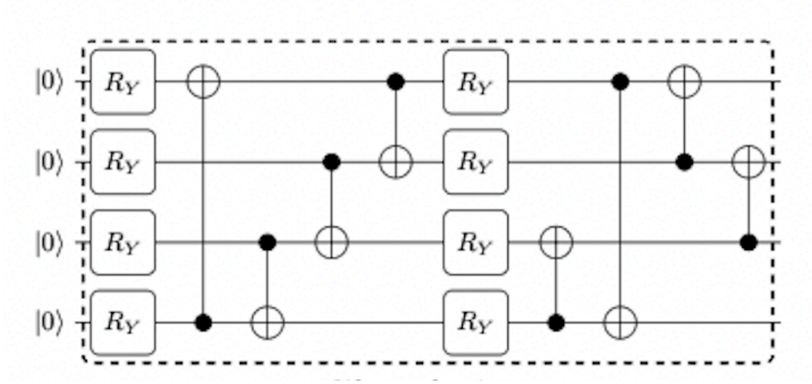
Various VQCs …

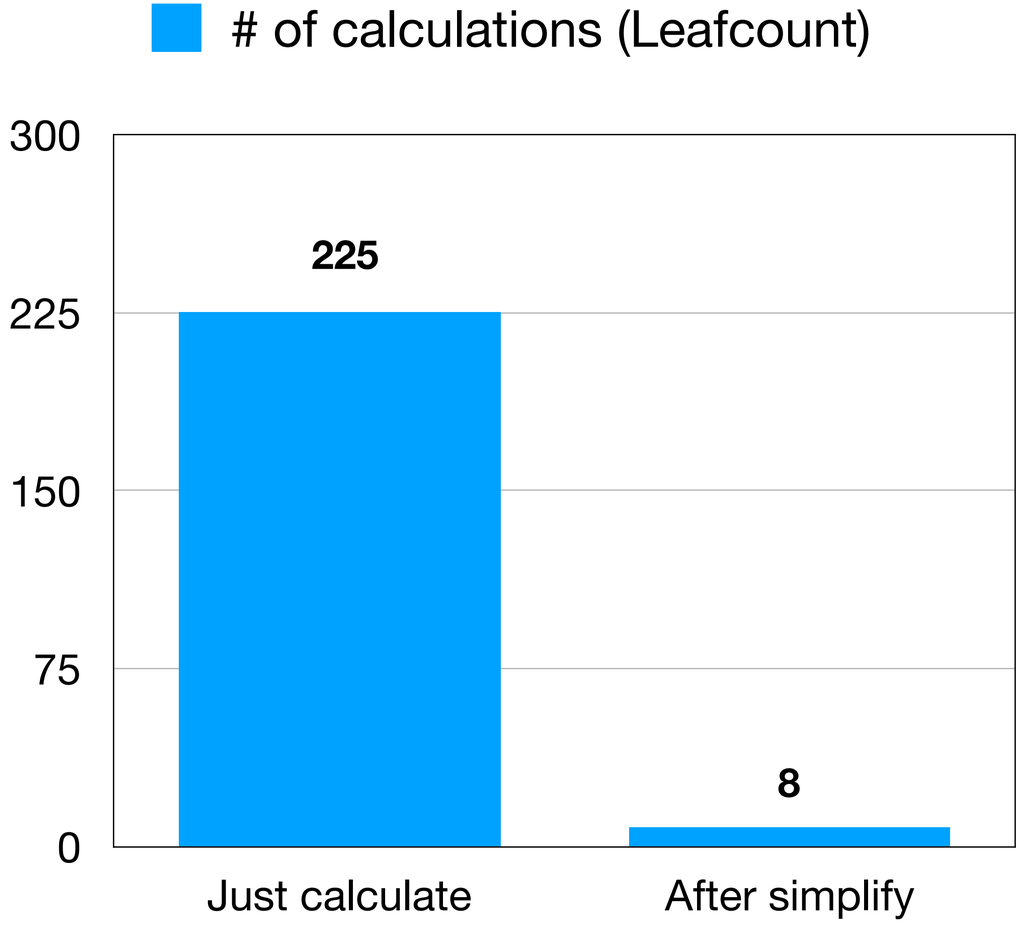




N=4, # of gates = 8, # of layers = 1

Various VQCs …

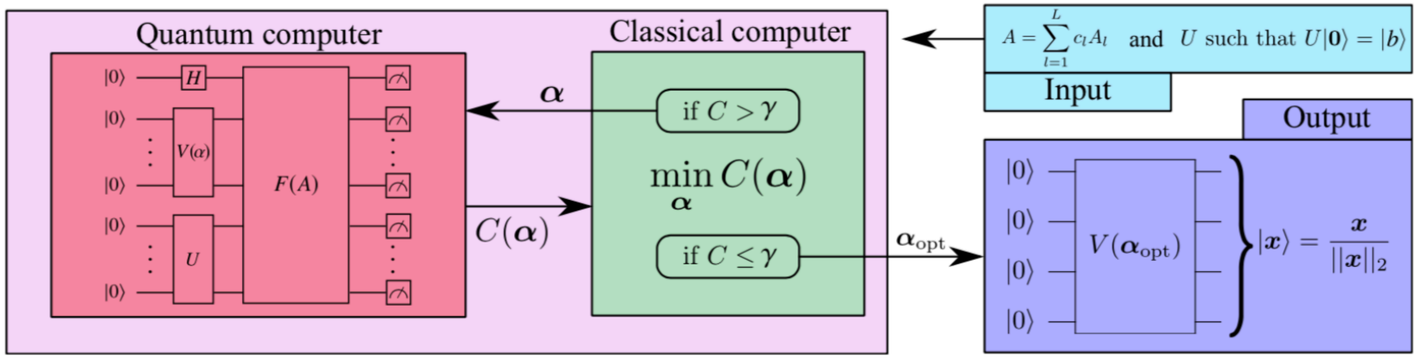




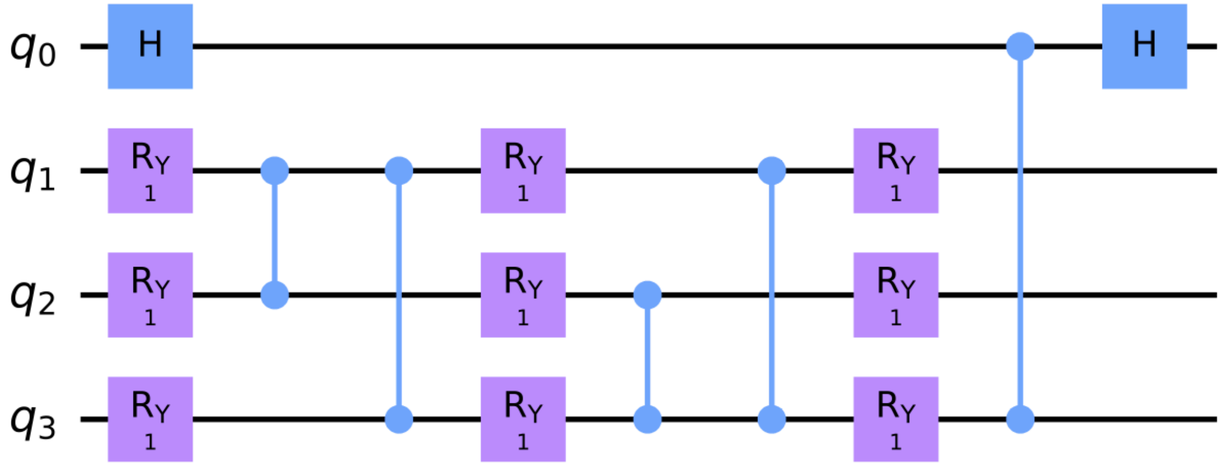
3. Application in Variational Quantum Linear Solver (VQLS)

VQC for Variational Quantum Linear Solver (VQLS)

The Variational Quantum Linear Solver, or the VQLS is a variational quantum algorithm that utilizes VQE in order to solve systems of linear equations more efficiently than classical computational algorithms. Specifically, if we are given some matrix $\mathbf{A}$, such that $\mathbf{A}|\mathbf{x}\rangle = |\mathbf{b}\rangle$, where $|\mathbf{b}\rangle$ is some known vector, the VQLS algorithm is theoretically able to find a normalized $|x\rangle$ that is proportional to $|\mathbf{x}\rangle$, which makes the above relationship true.

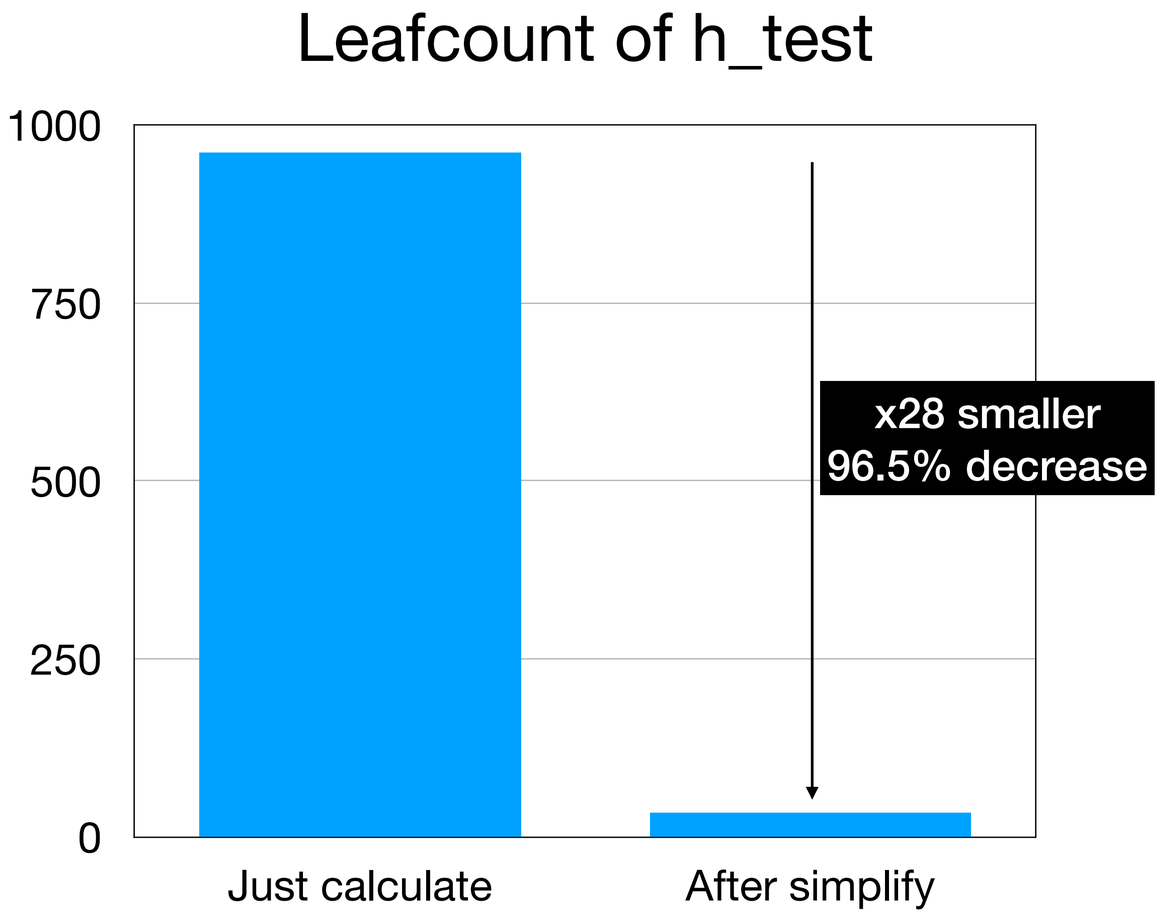


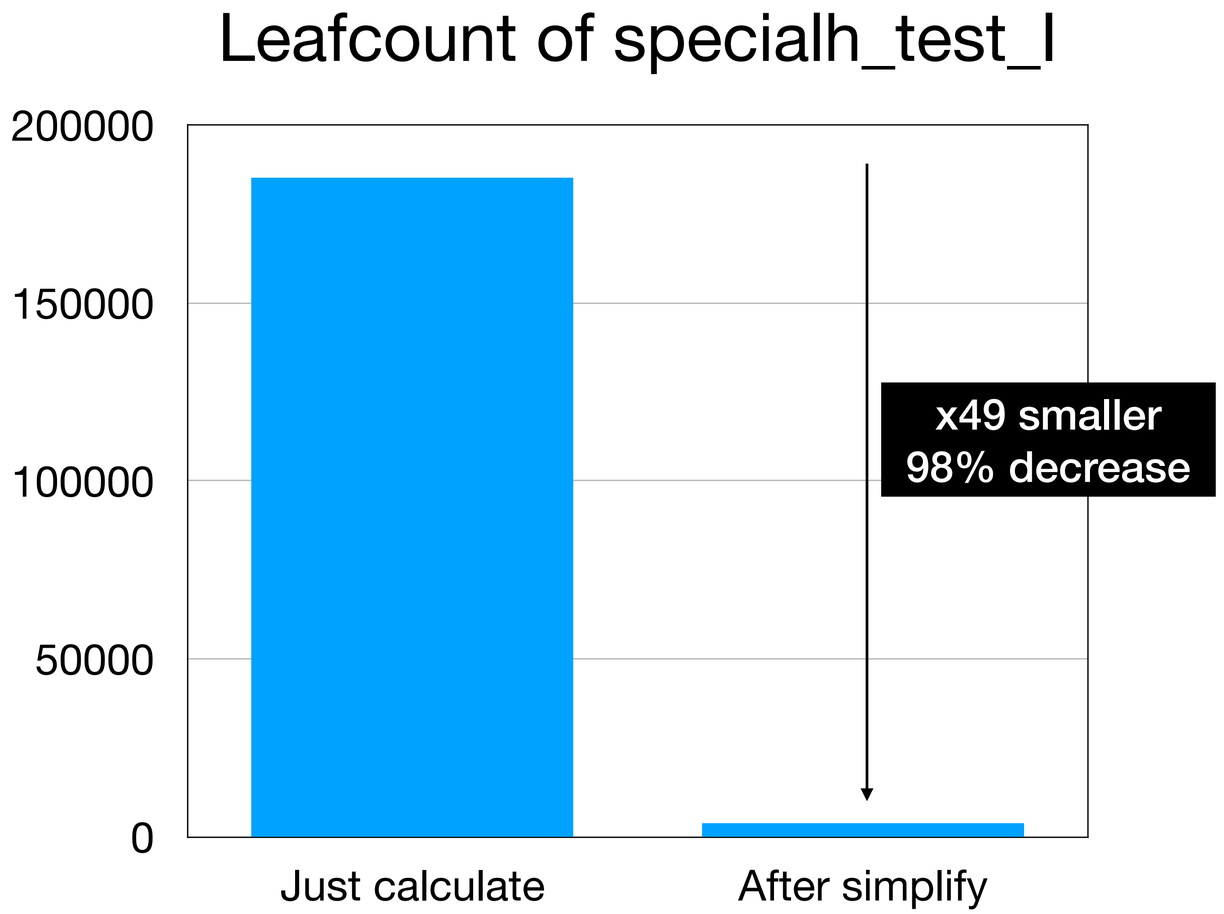
Objective: Optimizing the parameters of 9 Ry gates, minimizing P(q0; 0)



Change the parameters of Ry gate, in order to minimize the probability of 1st qubit in |0> state.

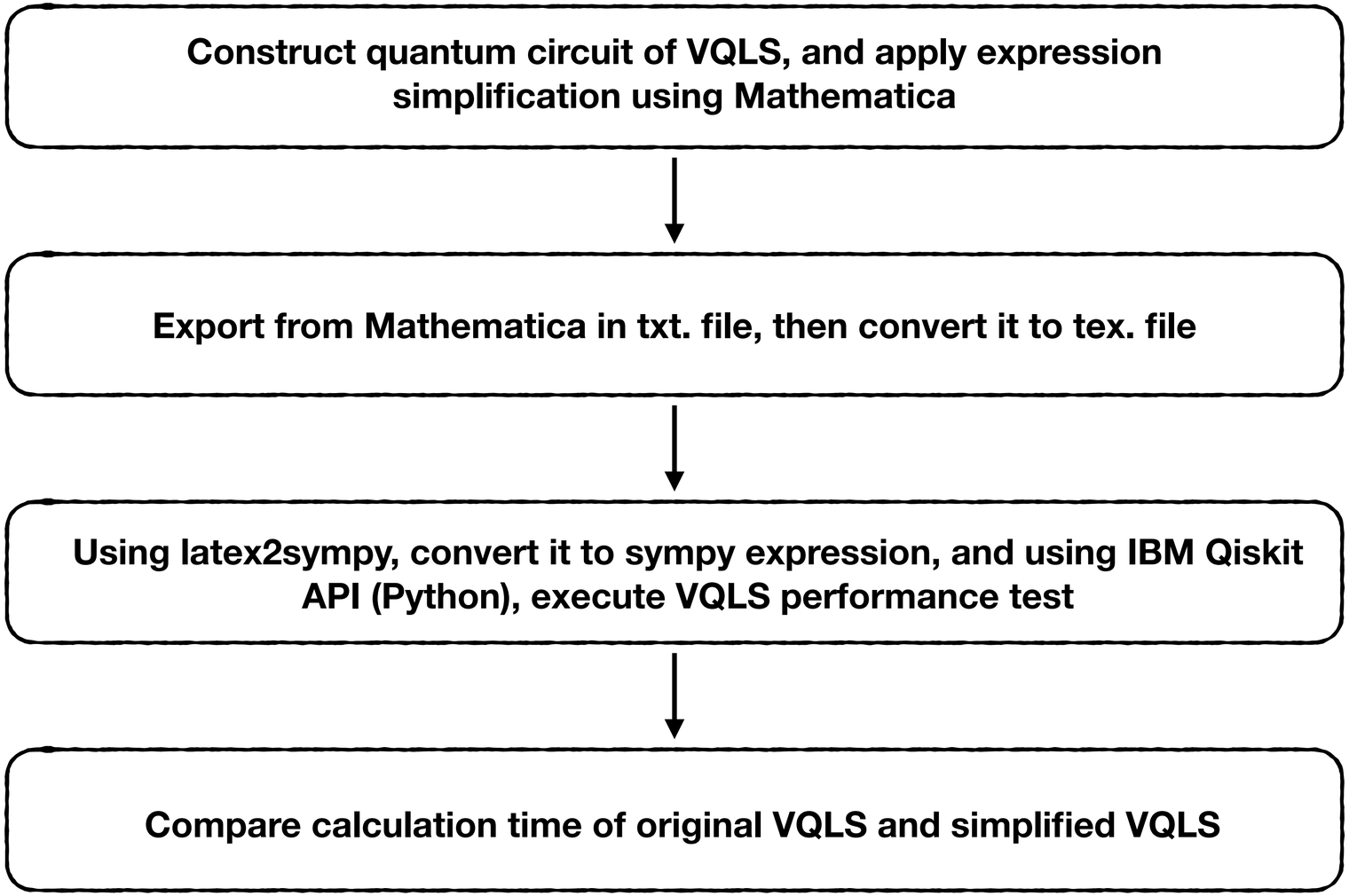
After simplify, decrease of expression size





Expecting x42 calculation time decrease after simplification

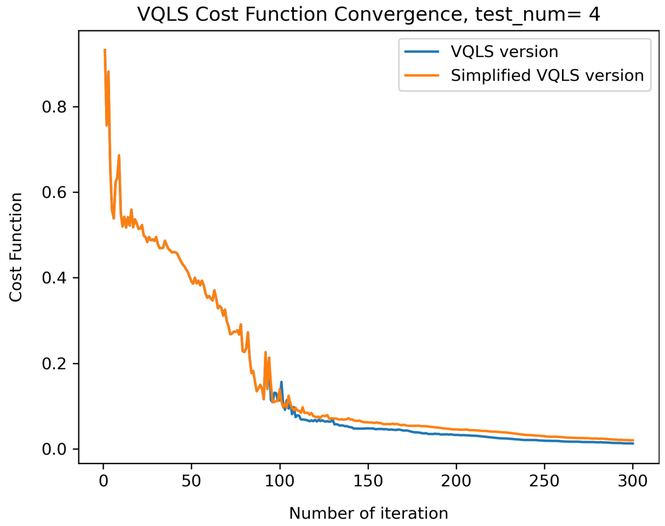
VQLS Simplification Overall Procedure

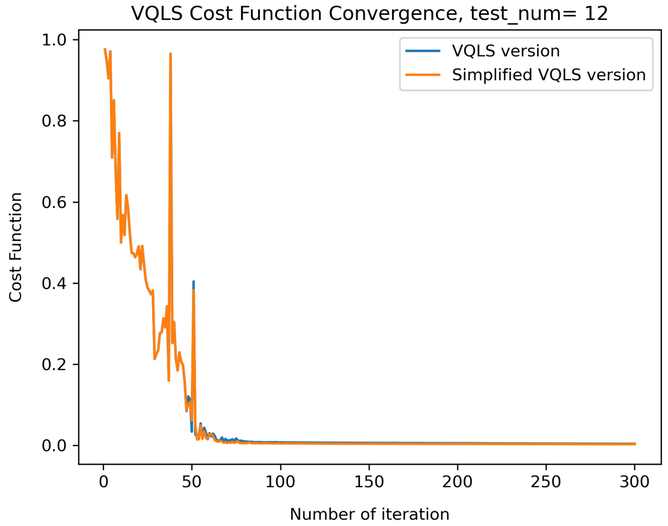


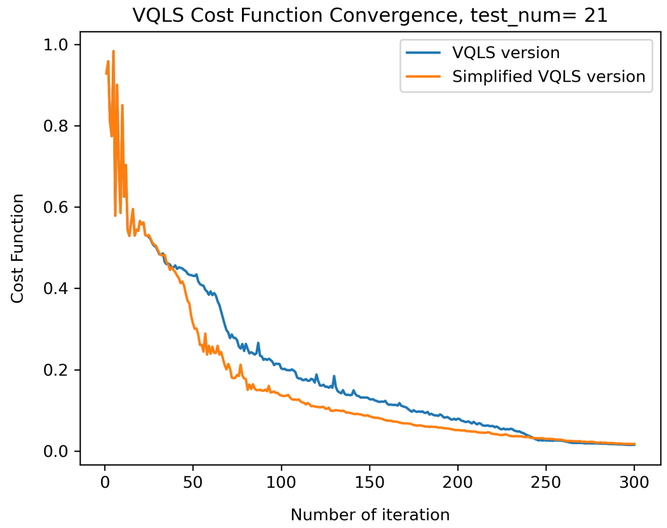
VQLS performance.py result

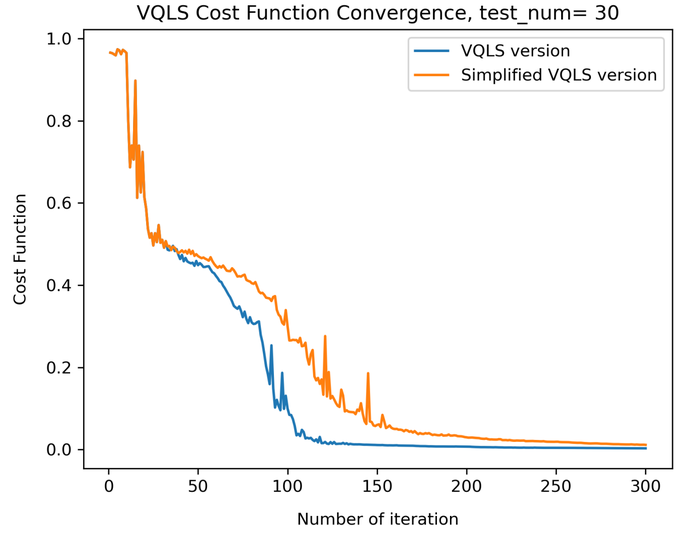
```
======VQLS RESULT======

     fun: 0.037080231501157024
   maxcv: 0.0
 message: 'Maximum number of function evaluations has been exceeded.'
    nfev: 200
  status: 2
 success: False
       x: array([1.62229185, 3.15382463, 0.33630285, 0.55690028, 2.95054901,
       0.19549135, 2.66144687, 2.27454045, 2.56457522])

[-0.05840416+0.j -0.05543708+0.j -0.04951071+0.j -0.05152622+0.j
 -0.35386427+0.j -0.3442107 +0.j -0.61076638+0.j -0.60964983+0.j]
VQLS Result:  (0.9629197685029154-0j)
VQLS Calculation time (s) :  88.06501197814941

======Short Custom VQLS RESULT======

     fun: 0.047928627327095574
   maxcv: 0.0
 message: 'Maximum number of function evaluations has been exceeded.'
    nfev: 200
  status: 2
 success: False
       x: array([1.68289628, 3.15280172, 0.3554633 , 0.61845431, 2.94279301,
       0.22906824, 2.67023508, 2.36629291, 2.5438394 ])

[-0.05867427+0.j -0.055442  +0.j -0.05229948+0.j -0.05248247+0.j
 -0.32647599+0.j -0.31834115+0.j -0.62458698+0.j -0.62446562+0.j]
Short Custom VQLS Result:  (0.9520713726908776-0j)
Short Custom VQLS Calculation time (s) :  0.48812007904052734

Speedup :  180.4166961360293
```

Identical result, faster calculation





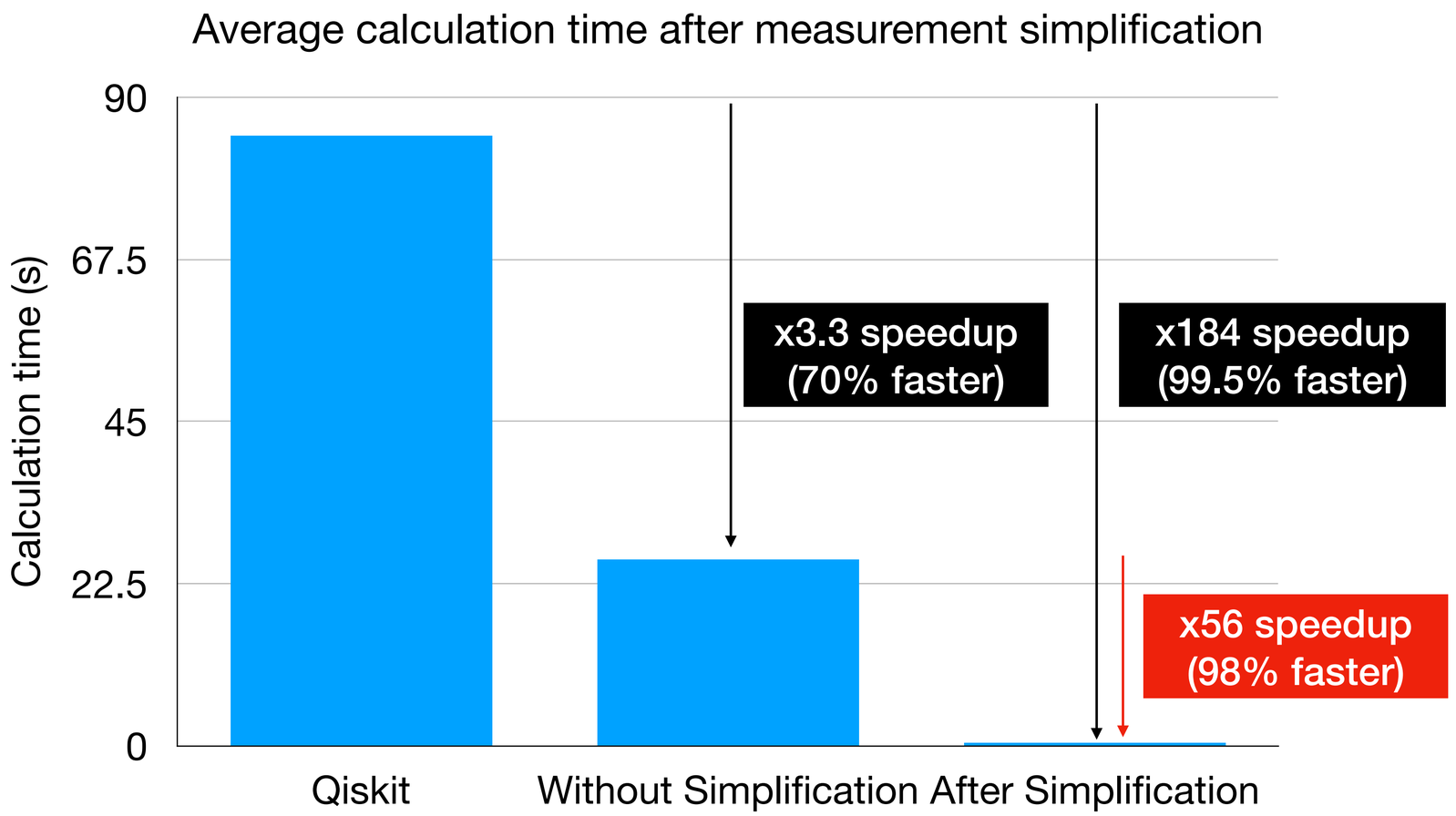





```
Total test_num:  100
Average VQLS time (s):  84.67696175813676
Average Simplified_VQLS time (s):  0.46015220403671264
Average Speedup:  184.01946359335685
```

```
Total test_num:  100
Average VQLS result:  (0.9820503674082908+0j)
Average Simplified_VQLS result:  (0.9827451118109928+0j)
Average VQLS time (s):  127.19085211038589
Average Simplified_VQLS time (s):  0.6870048308372497
Average Speedup:  185.1382208701196
```

Result of Simplification: Speedup in VQLS



Result of Simplification

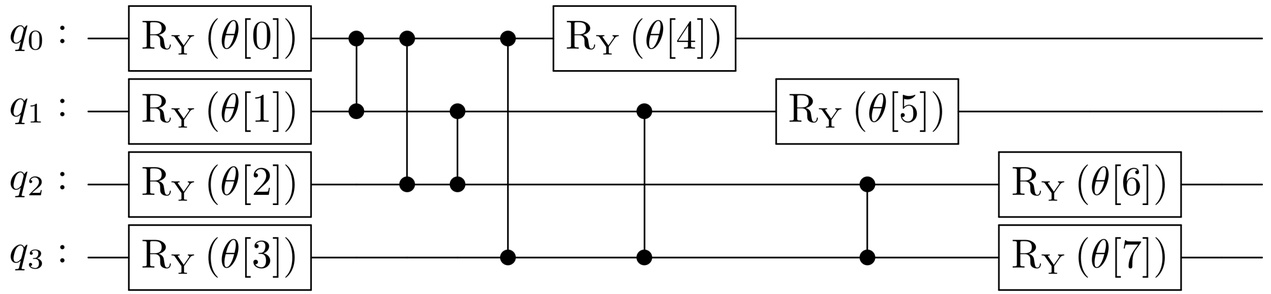
	Size (h_test)	Size (specialh_test)	Leafcount (h_test)	Leafcount (specialh_test)	Calculation Time (test_num=200)
Before Simplify	36KB	636KB	961	185256	25.9s
After Simplify	0.42KB	10KB	34	3790	0.46s
Improvement	85.7	63.6	28.2	48.9	56

4. Other applications in various Variational Quantum Circuit(VQC)s

Applications of VQC

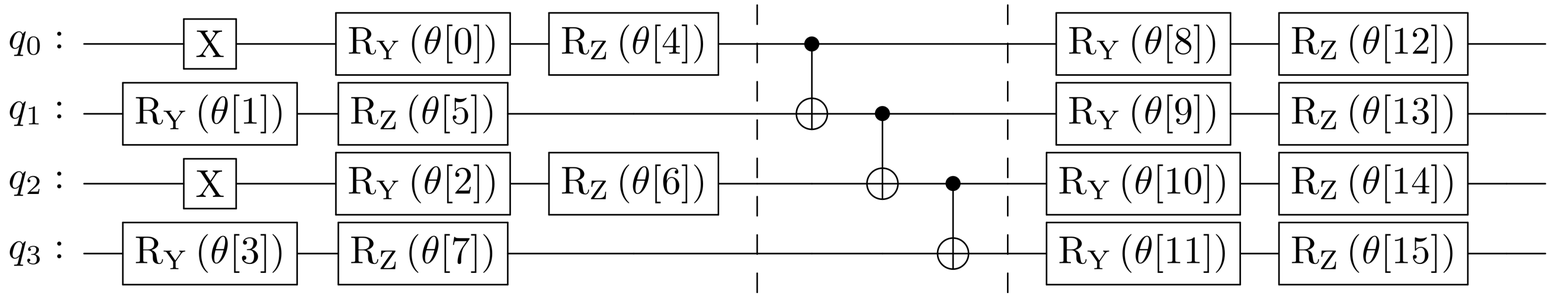
- Following problems can be solved with VQE (Variational Quantum Eigensolver), QAOA (Quantum Approximate Optimization Algorithm)
- Quantum Finance can be solved with (Portfolio optimization)
- Quantum Chemistry (Band gap calculation of OLED molecule)
- Quantum Machine Learning (Image Classification speed up)
- Quantum Optimization (Battery revenue optimization)

Quantum Finance: Portfolio Optimization



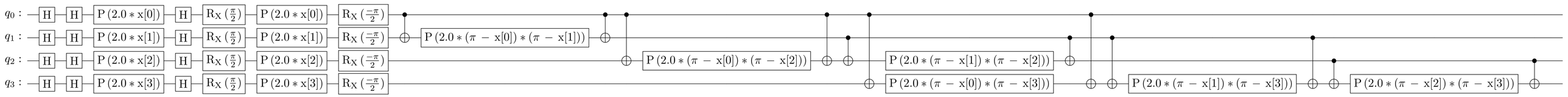
H	**Expression size (Before Simplification)**	**Expression size (After Simplification)**	**Ratio**
X_0	210468	19	11077
Y_0	210076	1	210076
Z_0	210468	20	10523
X_0X_1	210076	*	*
$X_0X_1X_2$	210468	59	3567
$X_0X_1X_2X_3$	209820	105	1998

Quantum Chemistry: Band gap calculation of OLED molecule



H	Expression size (Before Simplification)	Expression size (After Simplification)	Ratio
X_0	339732	89	3817
Y_0	339084	89	3810
Z_0	337940	97	3484
X_0X_1	340636	337	1011
$X_0X_1X_2$	342548	*	*
$X_0X_1X_2X_3$	343180	*	*

Quantum Machine Learning: Image Classification



# of dimension	Expression size (Before Simplification)	Expression size (After Simplification)	Ratio
5	1879400	53337	35

5. Discussion & Conclusion

Discussion & Conclusion

본 연구에서는 VQA를 사용하는 알고리즘에서 특정 큐빗 또는 기저의 진폭만이 필요한 경우 measurement simplification 이라는 과정을 통해 계산 속도와 메모리 사용량의 관점에서 speedup 이 있음을 살펴보았다. 여러 VQA 중에서 QDRL, VQLS, VQE, Quantum Kernel based QSVM 과 같은 모델들에서 사용된 양자회로에 simplification 을 적용시켜보았는데, 그 종류에 따라 3에서 크게는 10000까지도 expression size 가 줄어들었다. 현재 다양한 VQA가 계속 발표되고 있고, 이러한 알고리즘들은 NISQ에서 가까운 미래에 양자컴퓨터의 응용으로 사용될 수 있는 만큼 measurement simplification 방법은 광범위한 양자회로에 적용될 수 있는 방법이 될 것이다.

Measurement simplification이 직접적인 계산 속도에 영향을 주는지 확인하기 위해 VQLS 알고리즘에서의 효과를 살펴보았고, 그 결과 expression size 는 42배, calculation time 은 56배 줄어들었다. 이를 통해 표현식 간단화를 통해 나타나는 expression size 감소가 계산 속도에도 영향을 준다는 것을 확인했다. 이를 통해 iteration 횟수가 크고, dataset 이 큰 경우 simplification 이후 VQLS 와 같은 알고리즘을 실행시키는 것이 효과가 크게 나타날 것이라고 생각할 수 있다.

Quantum Finance, Quantum Chemistry, Quantum Machine Learning 에서 사용되었던 회로는 하나의 기저의 진폭만이 필요한 경우 이었는데, 이 경우 simplification 을 통한 expression 감소가 매우 컸다. 그 비율은 1000에서 10000까지 나타났는데, 이를 통해 simplified version 알고리즘들을 구현해볼 수 있을 것이고, 앞으로의 연구에서 이를 확인해 볼 수 있을 것이다.

더 연구가 필요한 부분은 simplification 이 이루어지는 조건이 무엇인지 이해하는 것이다. 각각의 VQC 구조마다 나타나는 improvement 가 다른데, 이가 어떤 조건에서 더 강하게 나타나는지 이해할 수 있다면, 더 효과적인 적용이 가능할 것이다. Mathematica를 이용한 Simplification 처리 자체가 시간이 오래 걸리는 경우가 있는데, 간단화되는 조건을 이해한다면 이러한 문제도 피할 수 있을 것이다.