NC STATE UNIVERSITY

# Quantum Enhanced Simulation Optimization
# - Quantum Amplitude Estimation -

Yunsoo Ha

# Simulation Optimization

- The general simulation optimization problem we consider it to find a setting of controllable parameters that minimizes a given objective function, i.e.,

$$\min_{y \in H} E[f(X,y)]$$

,where $X$ represents a sample path (simulation replication), $f$ is the sample performance measure.

- If $y$ is continuous, the expectation value $E[f(X,y)]$ can be evaluated as follows.
  - STEP 1: Load the probability $P_x$ into the amplitudes of $n$ qubits by operator $P_X$

$$\mathcal{P}_X |0\rangle_n = \sum_{x=0}^{N-1} \sqrt{p_\phi(x)} |x\rangle_n$$

  - STEP 2: Apply operator $F$

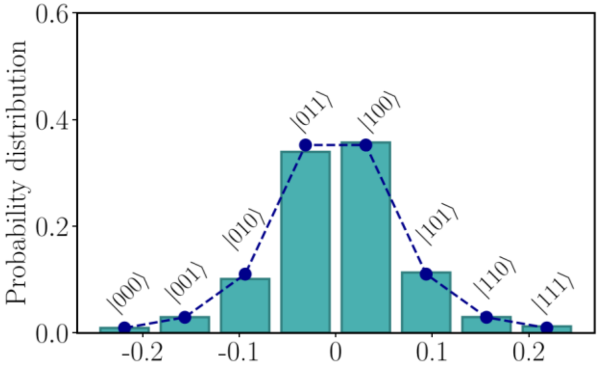
$$F|x\rangle|0\rangle = \sqrt{1-f(x)}|x\rangle|0\rangle + \sqrt{f(x)}|x\rangle|1\rangle$$

  - STEP 3: Apply Quantum Amplitude Estimation so that we can get the expectation value.

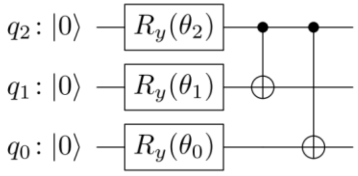
- Then, we can use the classical method in order to find the optimal solution $y^*$

# STEP 1: Load the probability $P_x$ into the amplitudes of $n$ qubits by operator $\mathcal{P}_X$

- We can load the bell-shaped probability distribution like normal distribution easily by using CNOT and $R_y$
- However, we cannot always easily load an exact representation of a generic data structure into an n-qubit state. Hence, researchers suggested qGAN which can learn a representation of the probability distribution underlying the data samples and load it into a quantum state.



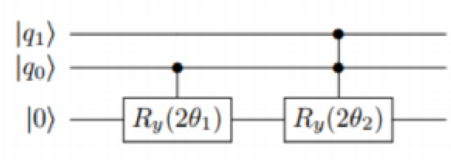

(a)




(b)

# STEP 2: Apply operator $F$

- We can use Multi-controlled $R_y(2\theta)$




$$
\begin{aligned}
|q_1\rangle |q_0\rangle |0\rangle &\rightarrow |q_1\rangle |q_0\rangle e^{-i\theta_2 q_0 q_1 Y} e^{-i\theta_1 q_o Y} |0\rangle && (1)\\
&= |q_1\rangle |q_0\rangle (\cos(\cdots)|0\rangle + \sin(\theta_2 q_0 q_1 + \theta_1 q_0)|1\rangle) && (2)\\
&= |q_1\rangle |q_0\rangle (\cos(p(q))|0\rangle + \sin(p(q))|1\rangle) && (3)
\end{aligned}
$$

,where $p(q) = \theta_2 q_0 q_1 + \theta_1 q_0$ is a polynomial.

- Then, exploiting the near linearity of the sine function when the argument is close to 0, we prepare $cp(q)$. Then, for $c \in \mathbb{R}$ small enough we have that

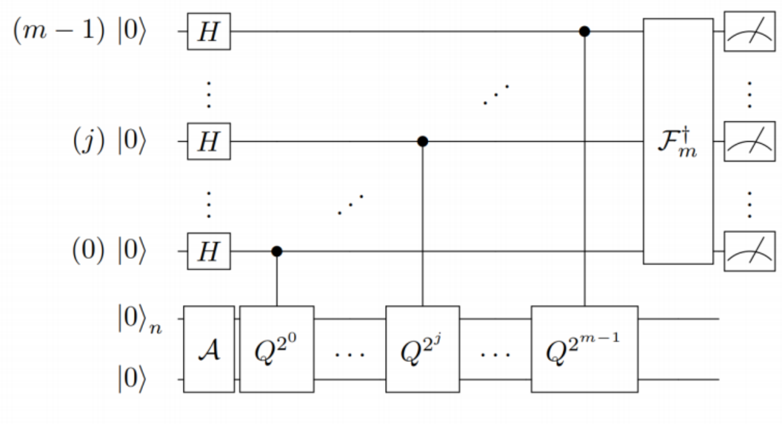
$$
|q_1\rangle |q_0\rangle (\cos(\cdots)|0\rangle + \sin(cp(q))|1\rangle) \approx |q_1\rangle |q_0\rangle (\cos(\cdots)|0\rangle + cp(q)|1\rangle)
$$

# STEP 3: Quantum Amplitude Estimation

- We prepared following state.

$$\mathcal{A}|0\rangle_n|0\rangle \to \sum_{i=0}^{N-1}\sqrt{1-f(i)}\sqrt{p_i}\,|i\rangle_n|0\rangle + \sum_{i=0}^{N-1}\sqrt{f(i)}\sqrt{p_i}\,|i\rangle_n|1\rangle \Leftrightarrow \cos(\theta_a)|i\rangle_n|0\rangle + \sin(\theta_a)|i\rangle_n|1\rangle$$

- QAE is a fundamental quantum algorithm with the potential to achieve a quadratic speedup for many applications that are classically solved through Monte Carlo (MC) simulation
- The canonical version of QAE is a combination of Quantum Phase Estimation(QPE) and Grover's Algorithm



# STEP 3: Iterative Quantum Amplitude Estimation

- Since $\sin^2(\theta_a)$ follows **Bernoulli distribution**, we can get the confidence interval with certain number ($n$) of measurement.
  - $\cos(\theta_a)\,|0\rangle + \sin(\theta_a)\,|1\rangle$

$$p - \frac{1.96\sqrt{p(1-p)}}{n} \leq \sin^2(\theta_a) \leq p + \frac{1.96\sqrt{p(1-p)}}{n}$$

  - $\cos(3\theta_a)|i\rangle_n|0\rangle + \sin(3\theta_a)|i\rangle_n|1\rangle$

$$p - \frac{1.96\sqrt{p(1-p)}}{n} \leq \sin^2(3\theta_a) \leq p + \frac{1.96\sqrt{p(1-p)}}{n}$$

$$\arcsin\left(\sqrt{p - \frac{1.96\sqrt{p(1-p)}}{n}}\right) \leq 3\theta_a \leq \arcsin\left(\sqrt{p + \frac{1.96\sqrt{p(1-p)}}{n}}\right)$$

# Example

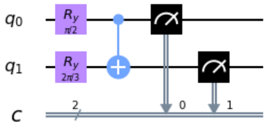
- we introduce simple example with 2-qubits and $f(x) = x^2$, where $x \in \{0, 1, 2, 3\}$.
- First, we prepare following bell-shaped probability distribution.
- Second, we apply operator $F$. In this case, we have for $q_i \in \{0, 1\},\ x = 2q_1 + q_0$. Hence,

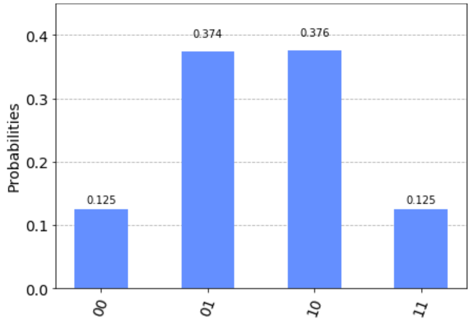
$$f(x) = f(2q_1 + q_0) = (2q_1 + q_0)^2 = 4q_1 + q_0 + 4q_1q_0$$

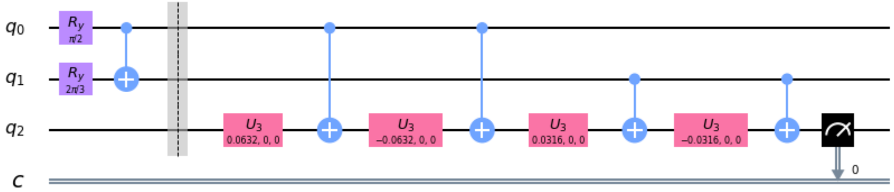
- As a result, we can have following state and when we measure last qubit, we can get the probability

$$\sum_{i=0}^{3} \sqrt{1 - cf(i)}\sqrt{p_i}\,|i\rangle_2\,|0\rangle + \sum_{i=0}^{3} \sqrt{cf(i)}\sqrt{p_i}\,|i\rangle_2\,|1\rangle \qquad P(1) = \sum_{i=0}^{3} cf(i)p_i = E(cf(x)) = cE(f(x))$$

,where $\mathrm{c} = 0.001$.

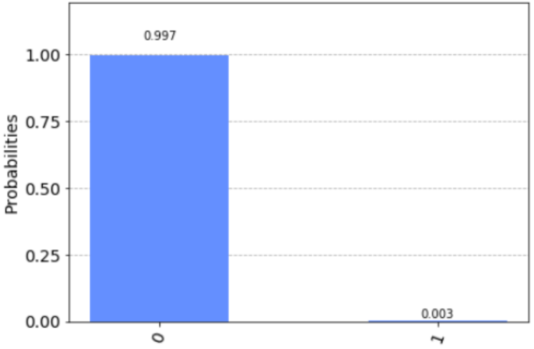


# Result

```
backend = BasicAer.get_backend('qasm_simulator')
shots = 100000
results = execute(qo, backend=backend, shots=shots).result()
answer = results.get_counts()
plot_histogram(answer)
```

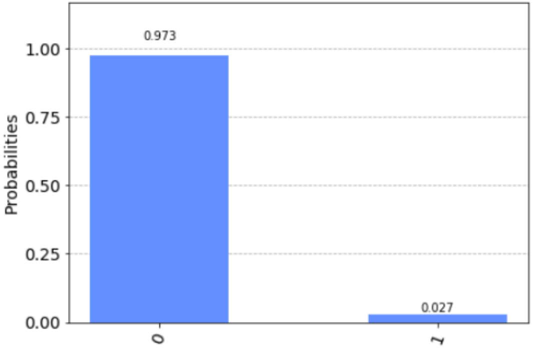



```
probability = answer['1']/shots
Approximate_answer = probability/cost
```

```
print('Exact value:     \t%.4f' % exact_answer)
print('Estimated value:\t%.4f' % Approximate_answer)
```

```
Exact value:        3.0024
Estimated value:    2.9700
```

```
backend = BasicAer.get_backend('qasm_simulator')
shots = 50000
results = execute(qo, backend=backend, shots=shots).result()
answer = results.get_counts()
plot_histogram(answer)
```




```
probability = answer['1']/shots
K = np.arcsin(np.sqrt(probability)) # theta*(2k+1)
k = K/(2*num_q+1) # theta
t = np.sin(k)
Approximate_answer = t**2/cost
```

```
print('Exact value:     \t%.4f' % exact_answer)
print('Estimated value:\t%.4f' % Approximate_answer)
```

```
Exact value:        3.0042
Estimated value:    2.9882
```

```
print('theta_lower:           \t%.4f' % theta_l)
print('theta_upper:           \t%.4f' % theta_u)
print('lower CI for E(x^2):  \t%.4f' % A_l)
print('upper CI for E(x^2):  \t%.4f' % A_u)
print('Number of measurement:\t%.4f' % N_measure)
```

```
theta_lower:            0.0538
theta_upper:            0.0554
lower CI for E(x^2):    2.8921
upper CI for E(x^2):    3.0625
Number of measurement:  10000.0000
```