PASQAL

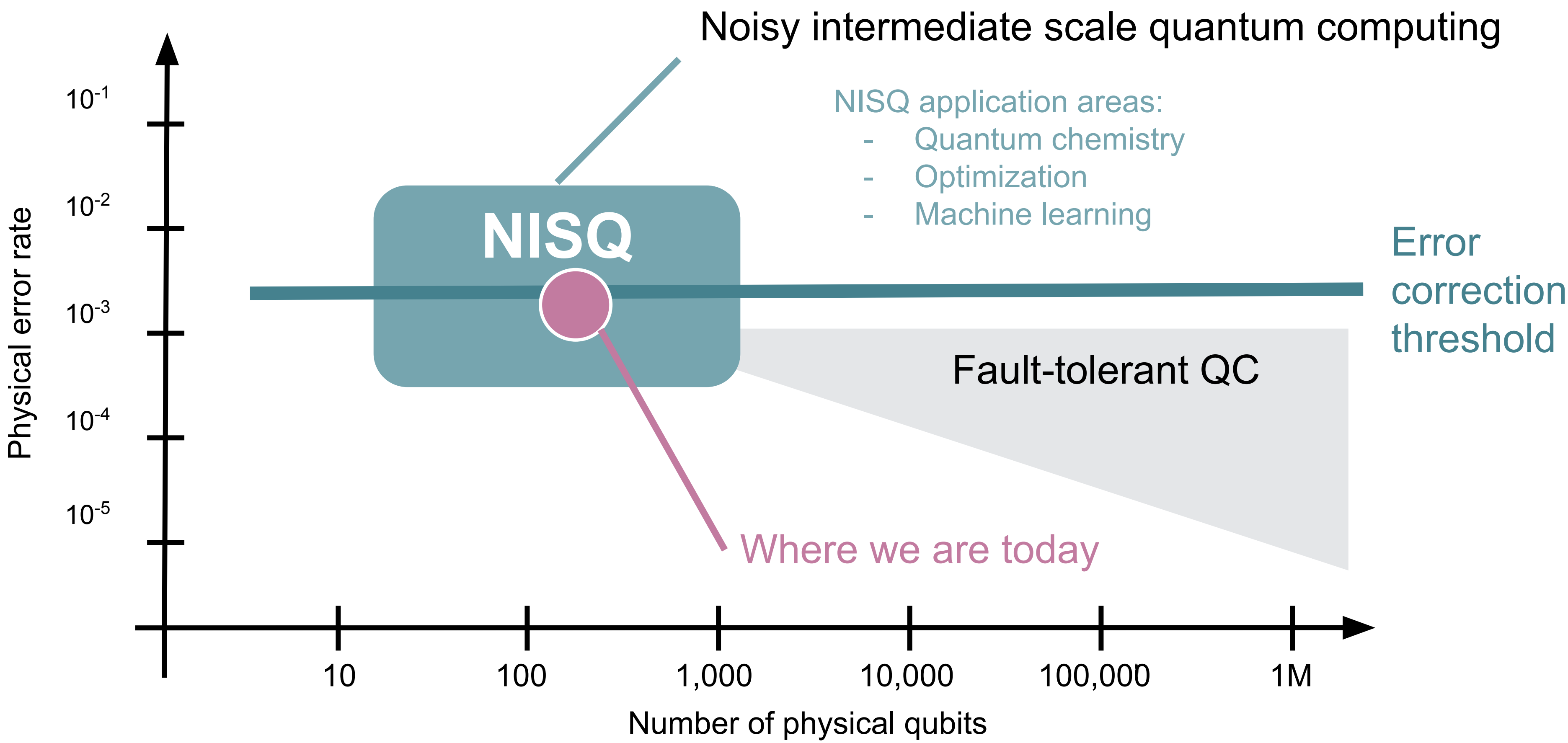
QUANTUM DISCOVERY

The quantum approximate optimization algorithm (QAOA)

PASQAL
www.pasqal.com
office@pasqal.com
7 rue Léonard de Vinci
91300 Massy
France

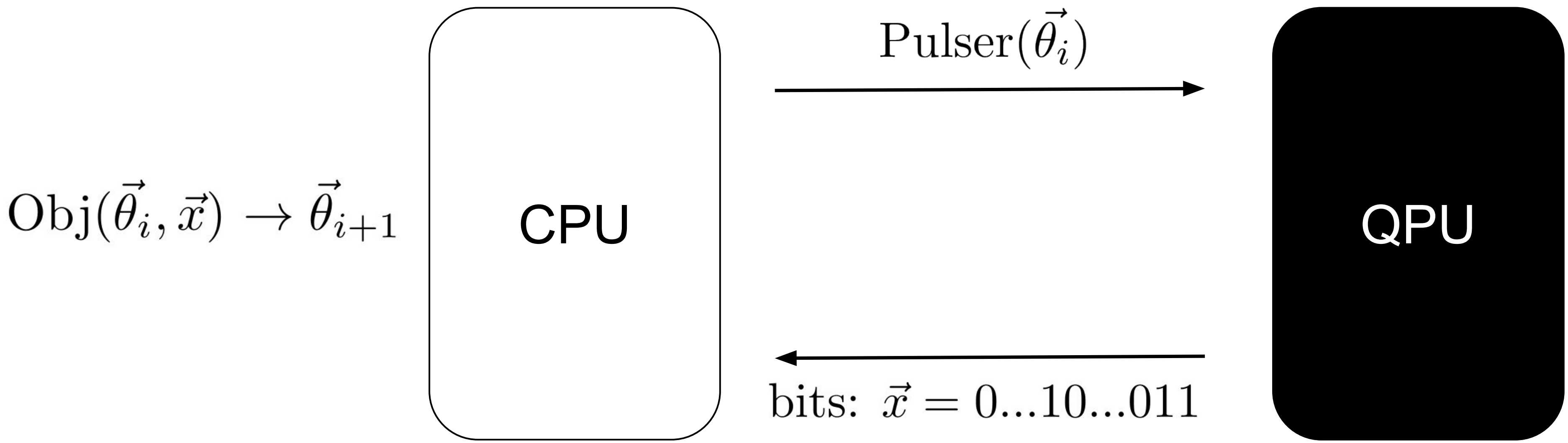
Hybrid quantum / classical computing

Quantum computing in the NISQ era



Hybrid quantum / classical computing

Variational quantum algorithms



The quantum approximate optimization algorithm

Algorithm description

A way to implement cost Hamiltonians is by adiabatic evolution:

$$H(t) = u(t)\underbrace{H_M}_{\text{Mixer Hamiltonian}} + (1 - u(t))\underbrace{H_C}_{\text{Cost Hamiltonian}}$$

In QAOA, starting in an eigenstate of the mixer Hamiltonian, we exponentiate and parametrize in *p* steps by *p* betas and *p* gammas:

$$U_{ansatz} = e^{-i\beta_p H_M} e^{-i\gamma_p H_C} \ldots e^{-i\beta_1 H_M} e^{-i\gamma_1 H_C}$$

$$\text{CPU} \rightleftarrows \text{QPU}$$

where the betas and the gammas are to be optimized with a classical optimizer.

The quantum approximate optimization algorithm

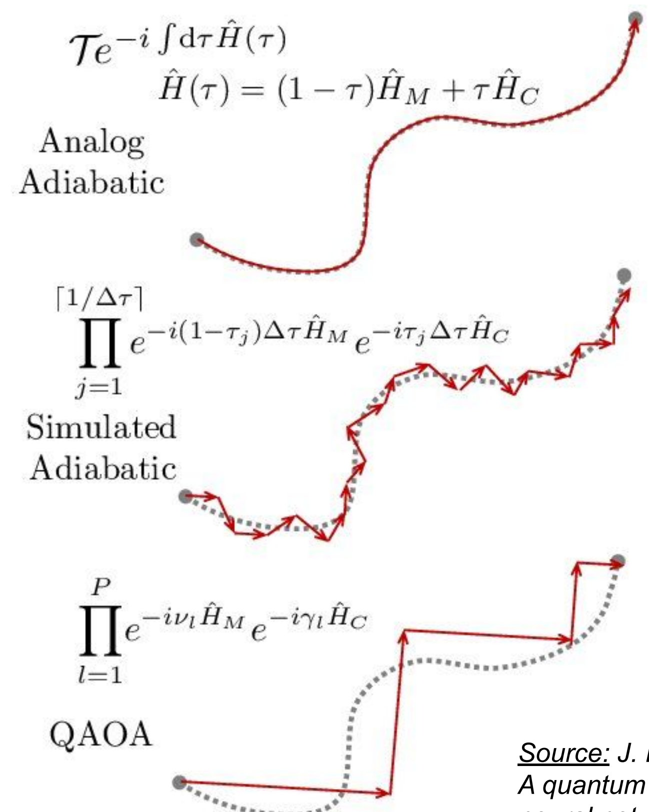
Connexion with the quantum adiabatic algorithm

- Runtime of a QAOA:

$$t_f = \sum_{i=1}^{p} (\gamma_i + \beta_i)$$

- (Discretized) control function:

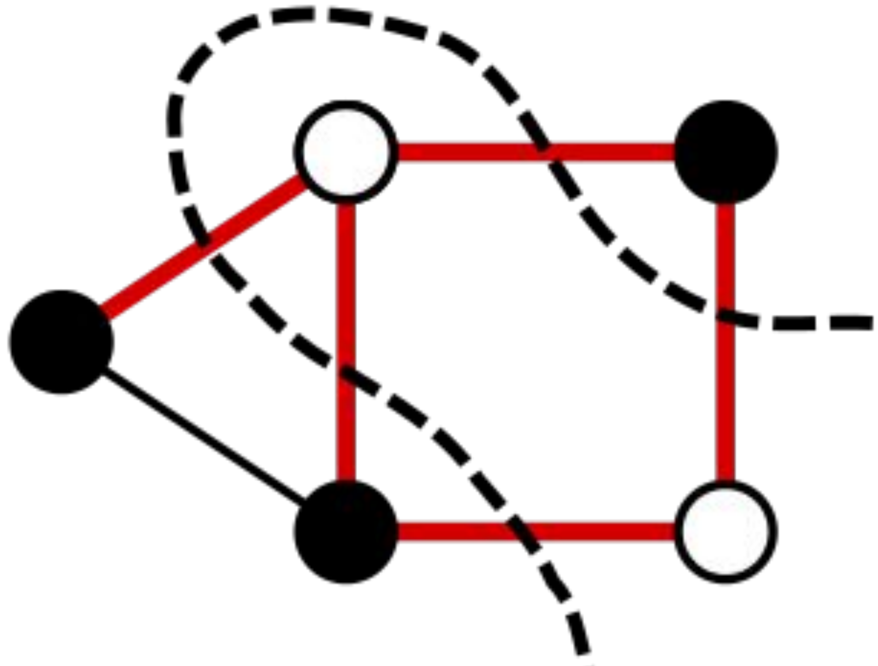
$$u_i = \frac{\beta_i}{\gamma_i + \beta_i}$$



<u>Source:</u> J. Biamonte et al. - A quantum algorithm to train neural networks using low-depth circuit

Solving the MaxCut problem with QAOA

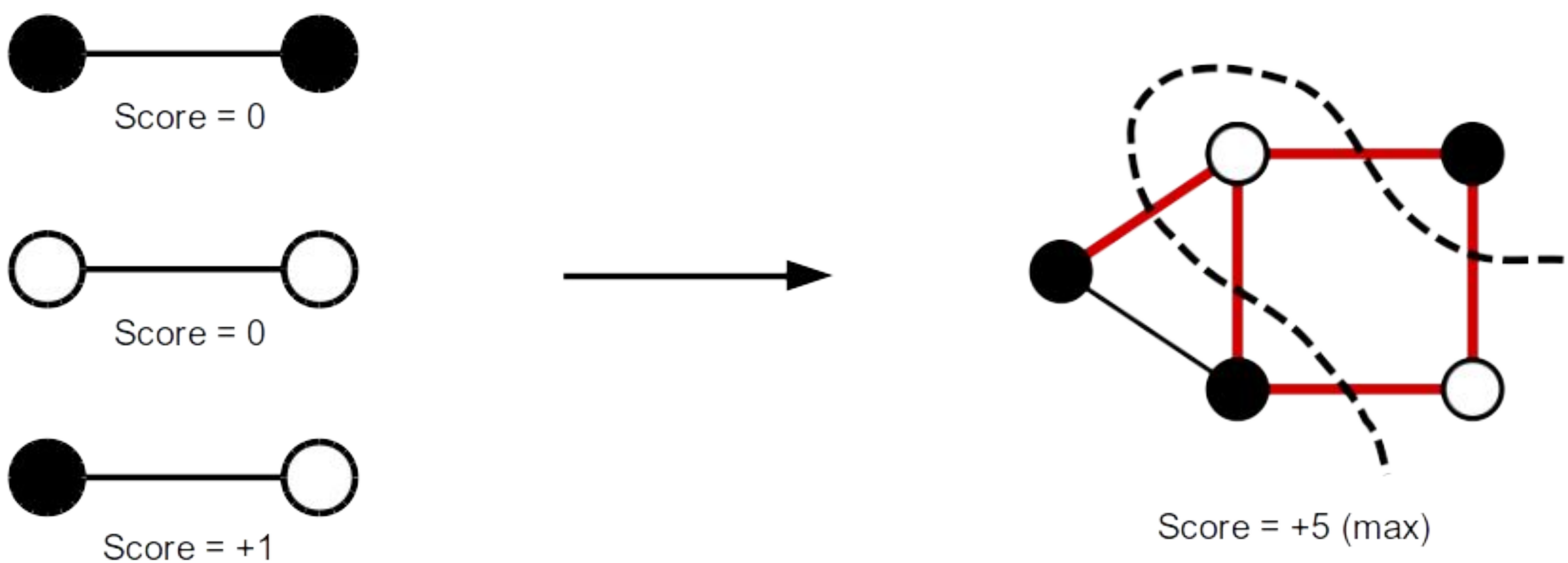
Problem statement



Solving the MaxCut problem with QAOA

Problem statement

$$\text{maximize:} \sum_{(i,j)\in E} x_i(1-x_j) \qquad x_i \in \{0,1\} \quad \forall i \in N$$



Solving the MaxCut problem with QAOA

Cost and mixer Hamiltonians

To apply QAOA, we first translate the cost function into an equivalent quantum cost Hamiltonian:

$$x_i \longrightarrow \frac{1 - Z_i}{2} \qquad\qquad C(x) \longrightarrow H_C = \sum_{(i,j)\in E} \frac{1}{2}(1 - Z_i Z_j)$$

Then we take the mixer Hamiltonian as a non-commutating operator with the cost Hamiltonian, a common choice is:

$$H_M = \sum_i X_i$$

Solving the MaxCut problem with QAOA

Cost and mixer Hamiltonians

- Unitary evolution with the cost Hamiltonian:

$$e^{-i\frac{\gamma}{2}(Id - Z_1 Z_2)} = e^{-i\gamma\begin{pmatrix}0&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&0\end{pmatrix}} = \begin{pmatrix}1&0&0&0\\0&e^{-i\gamma}&0&0\\0&0&e^{-i\gamma}&0\\0&0&0&1\end{pmatrix}$$

$$= \underbrace{\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{pmatrix}}_{\text{CNOT}_{12}} \cdot \underbrace{\begin{pmatrix}1&0&0&0\\0&e^{-i\gamma}&0&0\\0&0&1&0\\0&0&0&e^{-i\gamma}\end{pmatrix}}_{Id\otimes U_1(-\gamma)} \cdot \underbrace{\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{pmatrix}}_{\text{CNOT}_{12}}$$

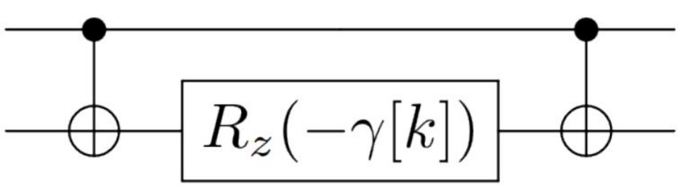
Solving the MaxCut problem with QAOA

Cost and mixer Hamiltonians

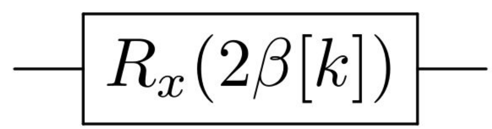
- Unitary evolution with the mixer Hamiltonian:

$$e^{-i\beta X} = e^{-i\beta\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}} = \begin{pmatrix}\cos(\beta) & -i\sin(\beta)\\ -i\sin(\beta) & \cos(\beta)\end{pmatrix}$$

→ Cost evolution (k-th layer):

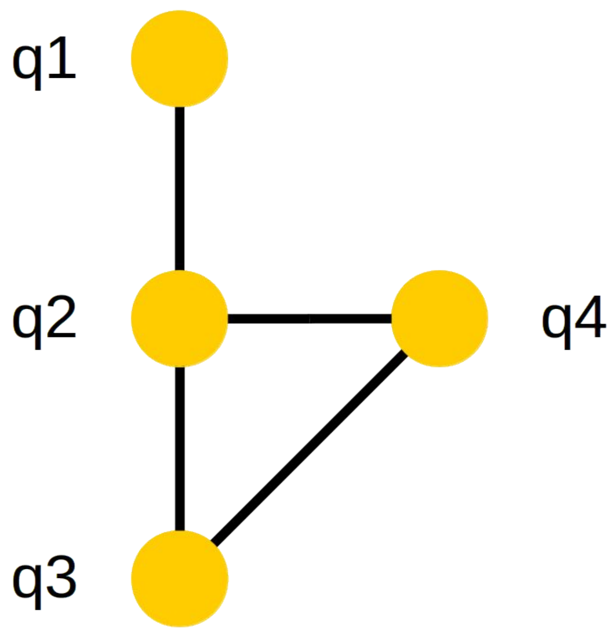


→ Mixer evolution (k-th layer):



Solving the MaxCut problem with QAOA

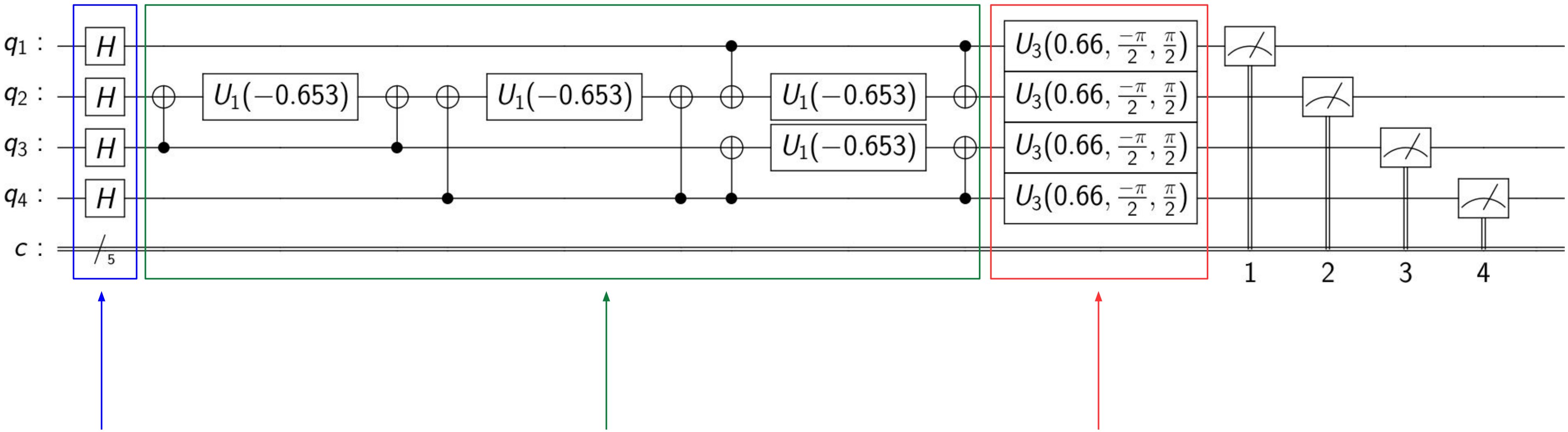
Example graph



MaxCut = 3

Solving the MaxCut problem with QAOA

One layer implementation



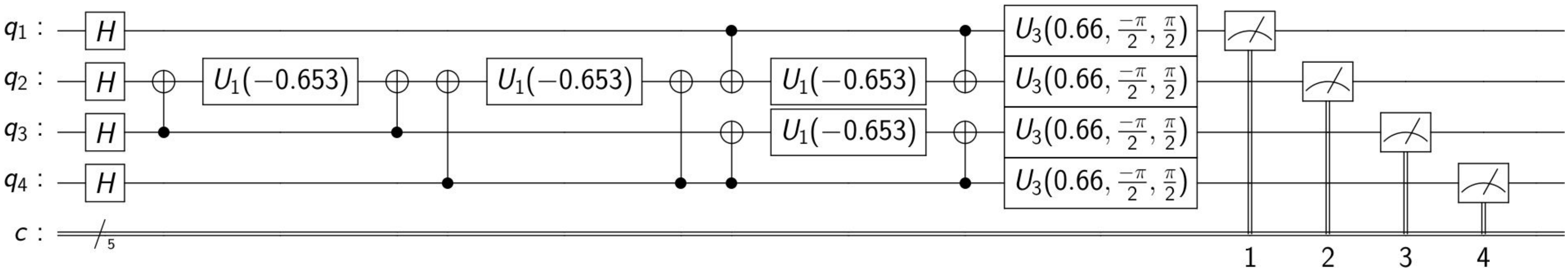
Initializing in an eigenstate of the mixer Hamiltonian

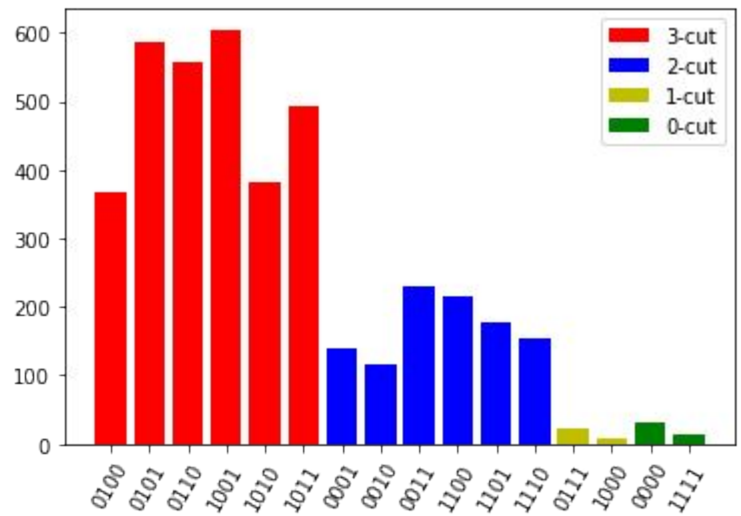
Evolution with the cost Hamiltonian for every edges in the graph

Evolution with the mixer Hamiltonian for every nodes in the graph

Solving the MaxCut problem with QAOA

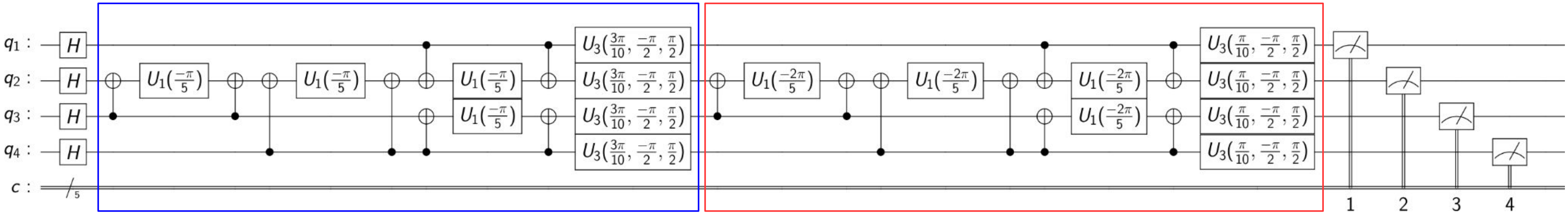
One layer implementation





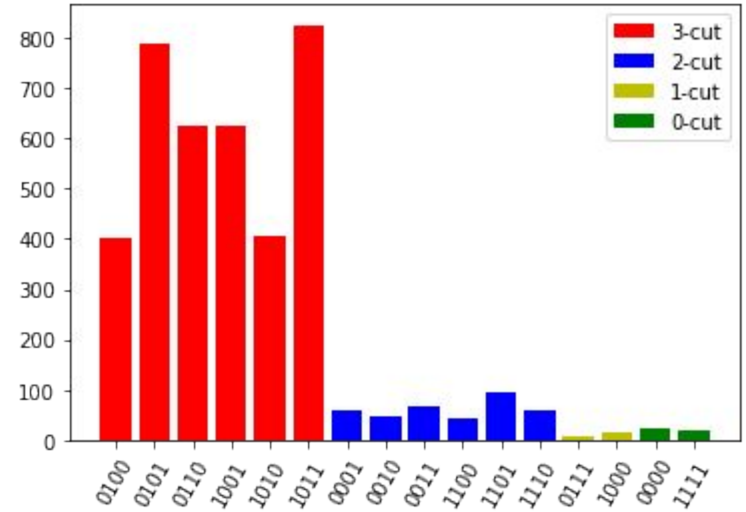
Solving the MaxCut problem with QAOA

Two layers implementation



1st layer

2nd layer



Conclusion

- ➔ Hybrid quantum / classical algorithms offer a better noise resilience in the NISQ era
- ➔ In hybrid setups, optimal parameters of a quantum algorithm are found using a classical optimizer and a cost function to minimize
- ➔ QAOA is an example variational algorithm to solve quadratic unconstrained binary optimization problems like MaxCut
- ➔ Adding layers in QAOA improves the quality of the solution at the price of an increased difficulty to find the optimal parameters
- ➔ Depending on the problem to solve, QAOA can be implemented in both analog and digital modes making this algorithm a versatile tool