Lecture 10 : VQE Review + QAOA

Last time we introduced Hamiltonians ~~as~~ as Hermitian matrices H that describe the general behavior of a physical system. We then talked about how we could use VQE to find the ground state of some system described by H. This ground state is the eigenvector(s) corresponding to the lowest eigenvalue of H. In a table:

Math representation	Physical Interpretation
Hermitian matrix H	~~Energy fields~~ Energy fields of a particular system
λ_0 smallest eigenvalue of H	The ground state energy
$\vec{\varepsilon}_0$ s.t. $H\vec{\varepsilon}_0 = \lambda_0 \vec{\varepsilon}$	The ground state of system described by H

Physical systems tend to want to be in lower energy states. Thus when they are sitting alone $\vec{\varepsilon}_0$ is the most "natural state" for the system to be in.

Ansatzes

These are the parameterized programs used in VQE. eg. $|\Psi(\vec{\theta})\rangle = U(\vec{\theta})|0\rangle$

One way to pick them is <u>variationally prepared programs</u> where we use the quantum computer to "simulate" the dynamics of a Hamiltonian $V(s)$ where

$$V(s) = A(s)\underset{\substack{\uparrow \\ \text{easy} \\ \text{hamiltonian}}}{H_0} + B(s)H_P \leftarrow \text{problem hamiltonian}$$

That acts like it is effectively slowly turning off H_0 and on H_P.

Our ansatz is then

$$U_{vp}(\vec{s}) = e^{-iV(s)/\hbar}|0\rangle$$

In order to run this on a quantum computer we need a way to simulate arbitrary Hamiltonians using our gates in discrete steps.

If we can write H as a sum of "small" (efficiently simulable) Hamiltonians H_i then we can write a program to simulate H efficiently.

<u>In general</u> if H_1 and H_2 are efficiently quantum simulable (aka they have a decomposition into a polynomial number of gates) then $H = H_1 + H_2$ is also efficiently simulable.

<u>Case (i)</u> H_1 & H_2 commute. Then $e^{-iH_1/t} e^{-iH_2/t} = e^{-iH_2/t} e^{-iH_1/t} = e^{-i(H_1+H_2)/t}$

and we can just apply one and then the other.

For example $H_1 = Z_1$ and $H_2 = X_0$

<u>Case (ii)</u> H_1 & H_2 do not commute. For example $H_1 = Z_1$ and $H_2 = X_1$

We then must use the Lie product formula:

$$e^{-i(H_1+H_2)/t} = \lim_{m \to \infty} \left(e^{-iH_1 t/m} e^{-iH_2 t/m} \right)^m$$

And in practice truncate for some finite m. In particular to bound the error $\leq \varepsilon$ takes $m = \mathcal{O}((\text{poly}(n)t)^2/\varepsilon)$

This is called <u>Trotterization</u>

Using higher approximations we can do better than a t^2 dependence. We can do it in $\mathcal{O}(t^{1+\delta})$ for and fixed $\delta > 0$ no matter how small. (Berry et al 2005) & (Childs 2004)

10.13

But wait! If I have $H = \sum H_i$ and it ~~is~~ is easy to simulate each H_i then can't I find the ground state of H easily?

The answer is no. It turns out that this problem is NP-hard.

Thm Finding the ground state of an arbitrary H is NP-hard.

Proof (~~Reduction~~ we can encode 3SAT into this problem)

Let $\varphi(x_1, \ldots, x_n) = c_1 \wedge \ldots \wedge c_m$ be a Boolean formula of n variables.

where each clause c_i acts on at most 3 variables.

Thus $c(b_i) = \{0, 1\}$ for $b_i \in \{0,1\}^3$

False True

Let $H = \sum H_i$ where $H_i = \sum_{b_j \in \{0,1\}^3} \neg c_i(b_j) |b_j\rangle\langle b_j|$

This encodes an energy "penalty" if any clause is violated. For example

for $c_i = v_1 \vee v_2 \vee v_3$

$$H_i = \begin{bmatrix} 1 & & & & & & & \\ & 0 & & & & & & \\ & & 0 & & & & & \\ & & & 0 & & & & \\ & & & & 0 & & & \\ & & & & & 0 & & \\ & & & & & & 0 & \\ & & & & & & & 0 \end{bmatrix}$$

Solving for the ground state of H would solve 3SAT!

$\lambda_0 = 0 \Rightarrow$ satisfiable φ

$\lambda_0 > 0$ otherwise

▷ We can also be inspired to choose states that are hard classically but we believe will be good models:

e.g. <u>Unitary coupled cluster</u> a unitary version of coupled cluster that is the gold standard classical method.

~~The idea is to explore~~ to some order k (where in practice <u>k=2</u> works well)

$$|\psi_{cc}^{(k)}(\vec{\theta})\rangle = e^{T^{(k)}(\vec{\theta})}|\Phi_R\rangle$$

⟵ initial guess or "reference state"

has $\mathcal{O}((3N)^k)$ real params and so is <u>quadratic</u> in params.

This can be turned into gates <u>efficiently</u> using, for example, Suzuki-Trotter.

▷ We can also choose an a-theoretical method and use a <u>hardware ansatz</u> (See $\boxed{\text{slides}}$ and [Kandala et al 2017])

<u>Solving for the optimal state</u>

Non-convex optimization. More to come in future lectures.

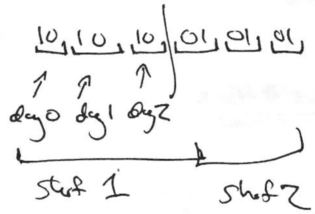
In the meantime scipy.minimize has Nelder-Mead for small examples.

Staff scheduling

Let x_{nds} be binary variables where $n \in N$ nurses
$d \in D$ days
$s \in S$ shifts

when $x_{nds} = 1$ then nurse n has been assigned shift s on day d.

Thus a full schedule is (3 nurses, 2 shifts, 2 days) a bitstring $b \in \{0,1\}^{NDS}$



We then apply constraints:

(A) 1 shift per day per nurse

$$\sum_s x_{nds} = 1 \quad \forall n, d$$

(B) A shift coverage matrix R_{ds} says how many nurses are needed on each shift on each day.

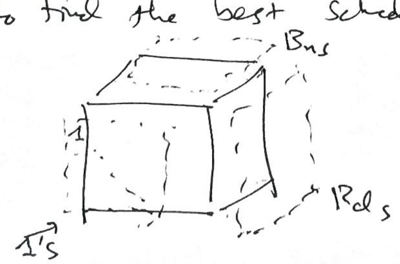
$$\sum_n x_{nds} = R_{ds}$$

(C) The balanced roster constraint specifies the # of times each nurse can work a certain shift. e.g. a limit on how many night shifts.

$$\sum_d x_{nds} = B_{ns}$$

$$H = \left(1 - \sum_s x_{nds}\right)^2 + \left(R_{ds} - \sum_n x_{nds}\right)^2 + \left(B_{ns} - \sum_d x_{nds}\right)^2$$

Minimizing against H tries to find the best schedule. You can think of this as a cube sudoku.



Quantum Approximate Optimization Algorithm

[QAOA] Hybrid algorithm used for constraint satisfaction problems

pronounced : kwaah-waah

Given binary constraints:

$$z \in \{0,1\}^n$$

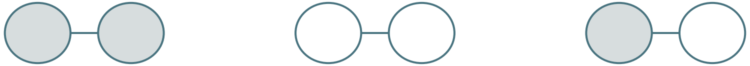
$$C_a(z) = \begin{cases} 1 & \text{if } z \text{ satisfies the constraint } a \\ 0 & \text{if } z \text{ does not .} \end{cases}$$

MAXIMIZE

$$\boxed{C(z) = \sum_{a=1}^{m} C_a(z)}$$

The MaxCut problem

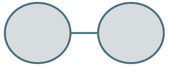
"Maximize disagreement on a colored graph"



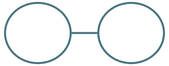
Score 0 Score 0 Score+1

The MaxCut problem

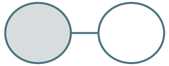
“Maximize disagreement on a colored graph”



Score 0

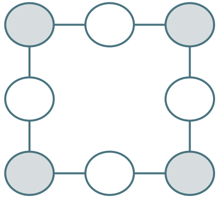


Score 0



Score+1

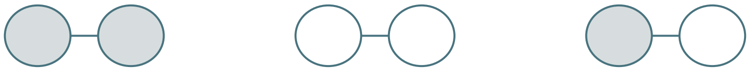
8-node “ring of disagrees”



Score = +8 (max)

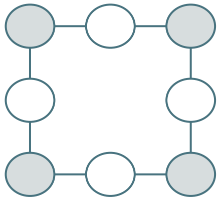
The MaxCut problem

“Maximize disagreement on a colored graph”



Score 0 Score 0 Score+1

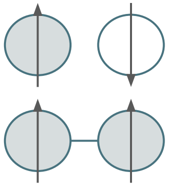
8-node “ring of disagrees”



Score = +8 (max)

MaxCut on a quantum computer

$$\hat{C}_{ij} = \frac{1}{2}\left(\mathbf{I} - \sigma_i^Z \sigma_j^Z\right)$$



Score+1

Score 0

Solving MAXCUT with QAOA

MAXCUT is an NP-complete problem

A quantum solver has *at most* a polynomial advantage for *exact* solution.

However, the Quantum Approximate Optimization Algorithm (QAOA) [Fahri et al, 2014] is a *heuristic approach* that has been shown to be competitive with the best classical algorithms.

There is a form of supremacy as well [Farhi & Harrow 2016]

QAOA: Farhi, Goldstone, and Gutman arXiv: 1411.4028 (2014)
MAXCUT & QAOA: Z. Wang et al, arXiv: 1706.02998 (2017)
Farhi & Harrow arXiv: 602.07674 (2016)

The Quantum Approximate Optimization Algorithm

Inspired by adiabatic quantum computing

$$\mathcal{H}(t) = (1-t)\mathcal{H}_\mathrm{i} + t\mathcal{H}_\mathrm{f}$$

"Driver" Hamiltonian

"Cost" Hamiltonian

Farhi et. al., 2014

The Quantum Approximate Optimization Algorithm

Inspired by adiabatic quantum computing

$$\mathcal{H}(t) = (1-t)\mathcal{H}_{\mathrm{i}} + t\mathcal{H}_{\mathrm{f}}$$

"Driver" Hamiltonian

"Cost" Hamiltonian

Discretize evolution into ***P*** steps

$$|\Psi_f\rangle = \prod_{p=1}^{P} e^{-i\beta_p \mathcal{H}_{\mathrm{D}}} e^{-i\gamma_p \mathcal{H}_C} |\Psi_i\rangle$$

Farhi et. al., 2014

The Quantum Approximate Optimization Algorithm

Inspired by adiabatic quantum computing

"Driver" Hamiltonian

$$\mathcal{H}(t) = (1-t)\mathcal{H}_{\mathrm{i}} + t\mathcal{H}_{\mathrm{f}}$$

"Cost" Hamiltonian

Discretize evolution into ***P*** steps

$$|\Psi_f\rangle = \prod_{p=1}^{P} e^{-i\beta_p \mathcal{H}_{\mathrm{D}}} e^{-i\gamma_p \mathcal{H}_C} |\Psi_i\rangle$$

P successive applications of *Cost* and *Driver* Unitaries

$$|\Psi_f\rangle = \prod_{p=1}^{P} U_{\mathrm{D}}(\beta_p) U_{\mathrm{C}}(\gamma_p) |\Psi_i\rangle$$

Farhi et. al., 2014

QAOA

The procedure

0. Prepare the initial state

$$|s\rangle = H^{\otimes n}|0...0\rangle$$

QAOA

The procedure

0. Prepare the initial state

$$|s\rangle = H^{\otimes n}|0...0\rangle$$

1. Apply the cost Hamiltonian

$$\mathcal{H}_C = \sum_{\langle jk\rangle} \frac{1}{2}(1 - \sigma_j^Z \sigma_k^Z)$$

2. Apply the driver Hamiltonian

$$\mathcal{H}_D = \sum_j \sigma_j^X$$

QAOA

The procedure

0. Prepare the initial state

$$|s\rangle = H^{\otimes n}|0...0\rangle$$

1. Apply the cost Hamiltonian

$$\mathcal{H}_C = \sum_{\langle jk \rangle} \frac{1}{2}(1 - \sigma_j^Z \sigma_k^Z)$$

2. Apply the driver Hamiltonian

$$\mathcal{H}_D = \sum_j \sigma_j^X$$

3. Exponentiate, parameterize in ***P*** steps by ***P*** Betas and ***P*** Gammas

$$U_{\text{ansatz}} = e^{-i\beta_p \mathcal{H}_D} e^{-i\gamma_p \mathcal{H}_C} ... e^{-i\beta_0 \mathcal{H}_D} e^{-i\gamma_0 \mathcal{H}_C}$$

QAOA

The procedure

0. Prepare the initial state

$$|s\rangle = H^{\otimes n}|0...0\rangle$$

1. Apply the cost Hamiltonian

$$\mathcal{H}_C = \sum_{\langle jk \rangle} \frac{1}{2}(1 - \sigma_j^Z \sigma_k^Z)$$

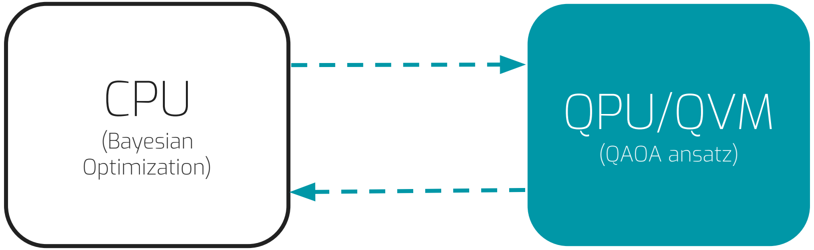
2. Apply the driver Hamiltonian

$$\mathcal{H}_D = \sum_j \sigma_j^X$$

3. Exponentiate, parameterize in ***P*** steps by ***P*** Betas and ***P*** Gammas

$$U_{\text{ansatz}} = e^{-i\beta_p \mathcal{H}_D} e^{-i\gamma_p \mathcal{H}_C} ... e^{-i\beta_0 \mathcal{H}_D} e^{-i\gamma_0 \mathcal{H}_C}$$

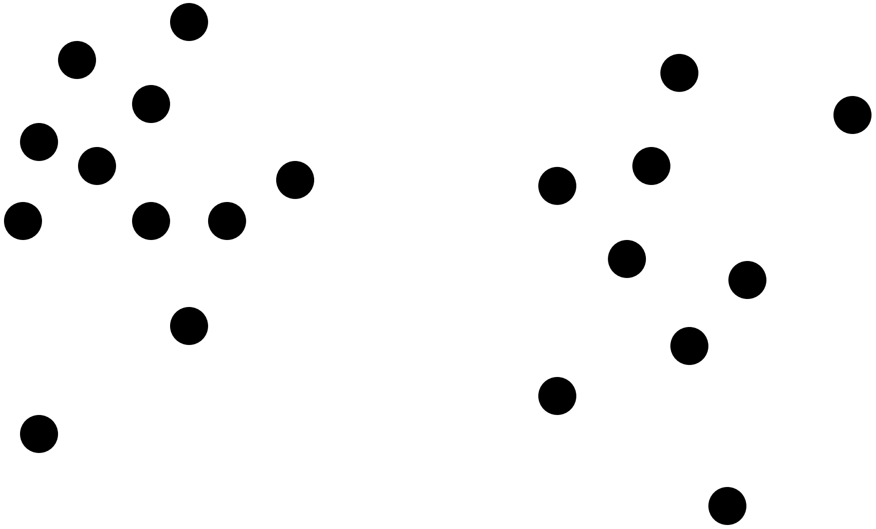
4. Optimize over betas and gammas



Staff Scheduling Problem (NP-complete)

See black board notes

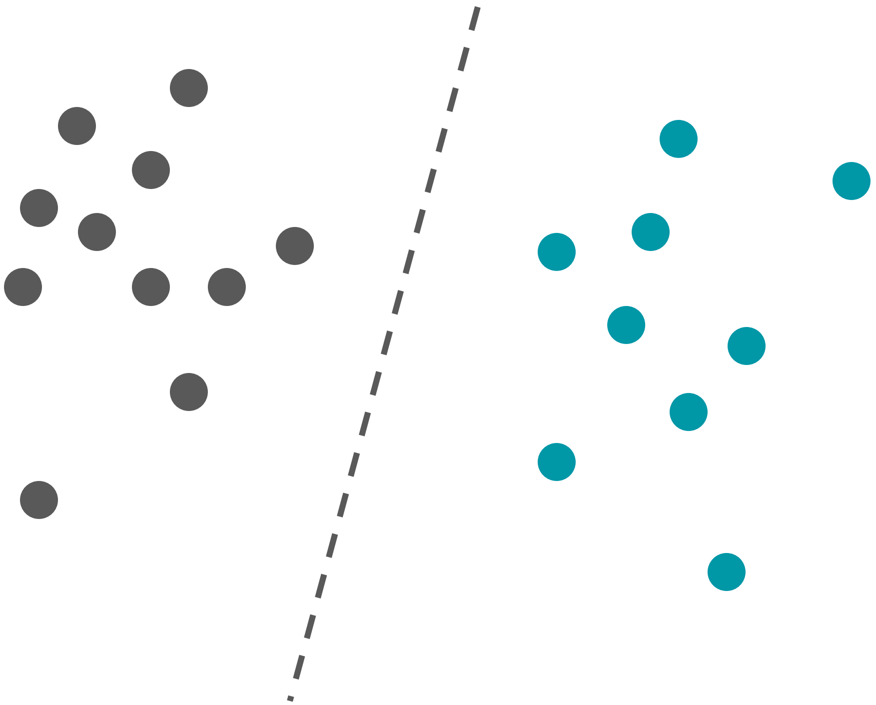
k-means clustering



> Given an unlabeled set of points

Otterbach, … WZ, ... et al 1712.05771 Unsupervised Machine Learning on a Hybrid Quantum Computer

k-means clustering



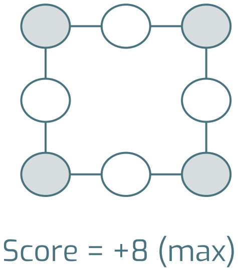
> Given an unlabeled set of points,

> find labels based upon ***similarity*** metric (e.g. Euclidean distance).

The MaxCut problem

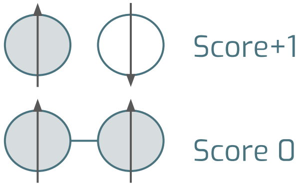
"Maximize disagreement on a colored graph"

8-node "ring of disagrees"



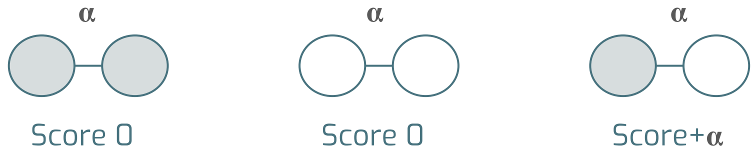
MaxCut on a quantum computer

$$\hat{C}_{ij} = \frac{1}{2}\left(\mathbf{I} - \sigma_i^Z \sigma_j^Z\right)$$

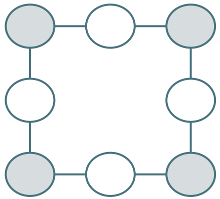


The weighted MaxCut problem

"Maximize disagreement on a colored graph"



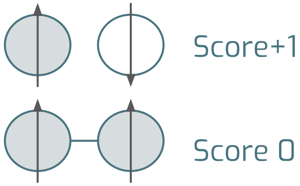
8-node "ring of disagrees"



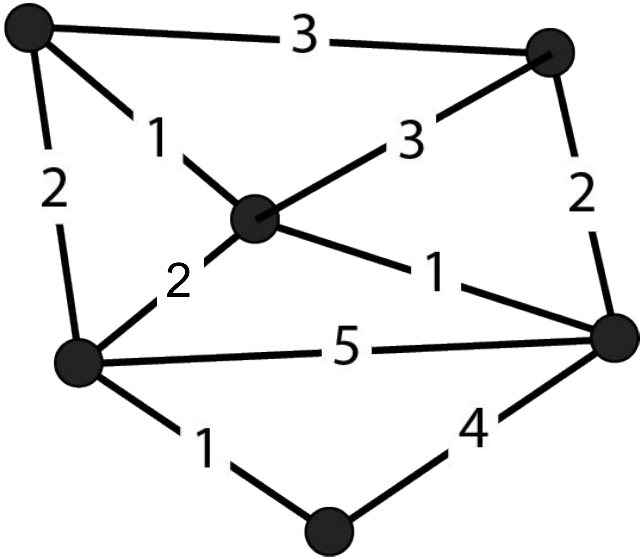
Score = sum(α_{ij})

MaxCut on a quantum computer

$$\hat{C}_{ij} = \frac{1}{2}\left(\mathbf{I} - \sigma_i^Z \sigma_j^Z\right) * \alpha_{ij}$$



2-means clustering as MAXCUT

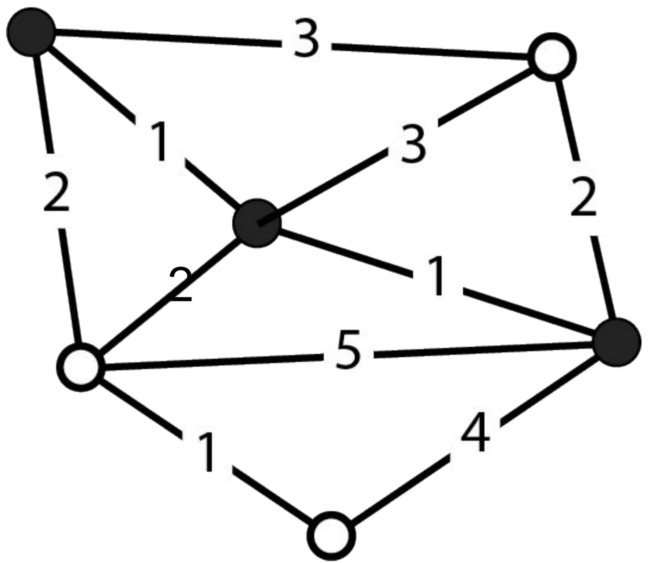


Construct a graph G=(V,E) where the edge weights w_i,j are determined by the distance metric.

Then, MAXCUT is a clustering algorithm for the original points.

$$\textsc{Maxcut} = \max_{\text{cut } S \subset E} \sum_{(i,j) \in S} w_{ij}$$

2-means clustering as MAXCUT



$$\text{MAXCUT} = \max_{\text{cut } S \subset E} \sum_{(i,j) \in S} w_{ij}$$

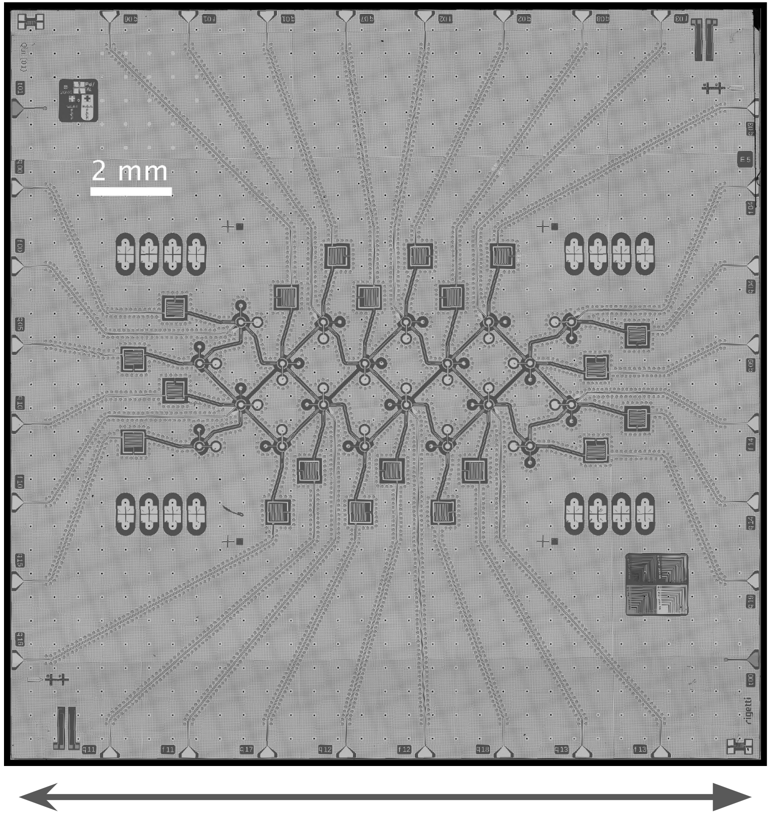
Construct a graph G=(V,E) where the edge weights w_i,j are determined by the distance metric.

Then, MAXCUT is a clustering algorithm for the original points.

Clustering transformed into an ***optimization*** problem.

Rigetti 19Q

Aluminum circuit on Silicon



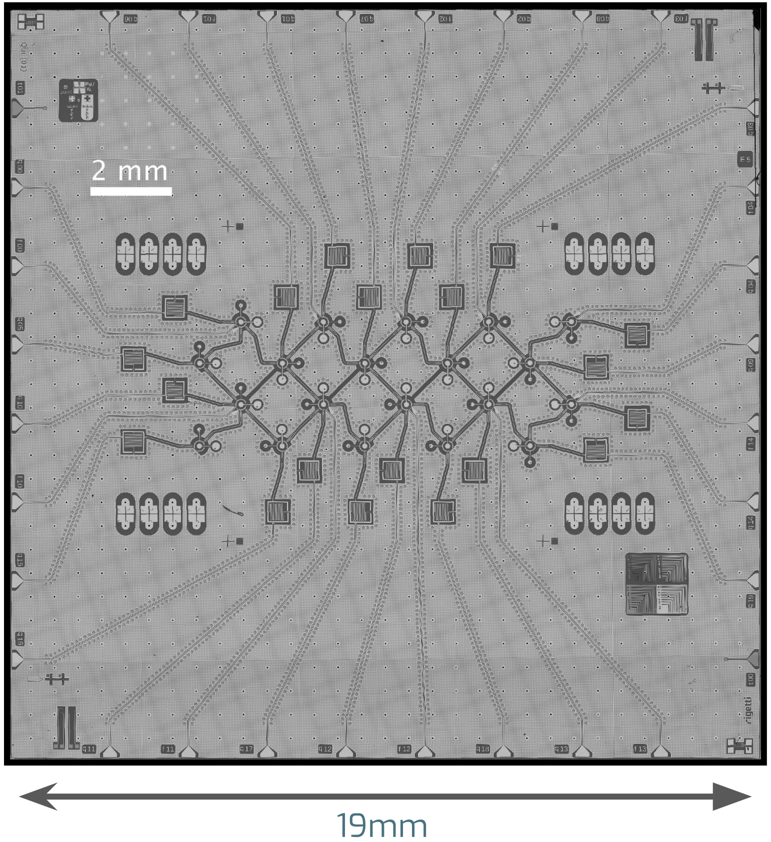
Device Properties

- 4x5 lattice of transmon qubits and quasi-lumped element resonators
- Fixed capacitive coupling between qubits
- Alternating arrangement of fixed-frequency and **tunable** (asymmetric) transmon qubits
- “19Q” because one tunable qubit was not tunable

Circuit QED:
Blais et al, PRA **69**, 062320 (2004)
Wallraff et al, Nature **431**, 162 (2004)
Hutchings et al, quant-ph/1702.02253 (2017)

Rigetti 19Q

Aluminum circuit on Silicon



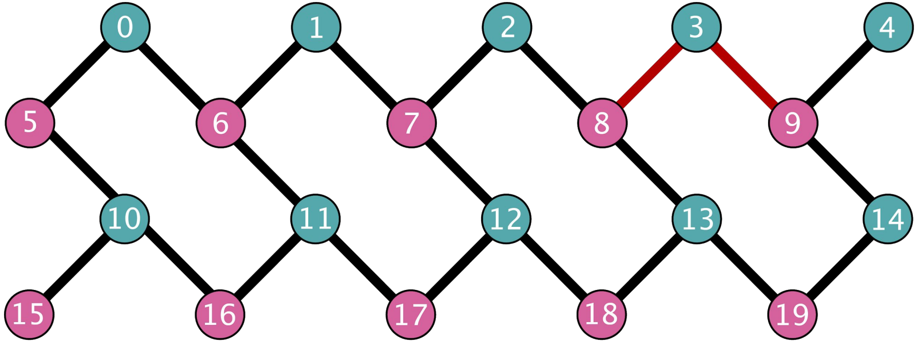
Device Properties

- 4x5 lattice of transmon qubits and quasi-lumped element resonators
- Fixed capacitive coupling between qubits
- Alternating arrangement of **fixed-frequency** and **tunable** (asymmetric) transmon qubits
- T_1 = 8-30 μs, T_2^* = 5-25 μs

Circuit QED:
Blais et al, PRA **69**, 062320 (2004)
Wallraff et al, Nature **431**, 162 (2004)
Hutchings et al, quant-ph/1702.02253 (2017)

Rigetti 19Q

19Q connectivity graph



Qubit-qubit interactions

- Fixed coupling between **fixed-frequency** and **tunable** (asymmetric) transmon qubits
- 2-qubit *parametric gates* use RF flux modulation to turn on effective resonance conditions
- Typical 2-qubit gate fidelity of 0.85-0.95

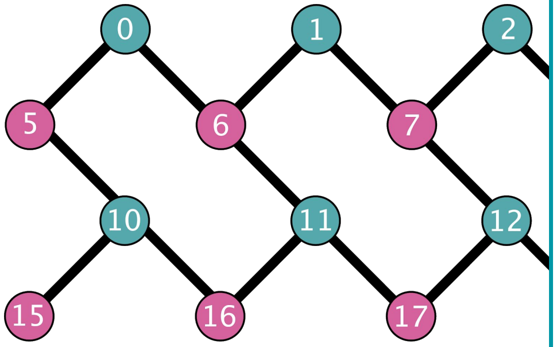
Parametric gates:
theory: N. Didier et al, arXiv:1706.06566 (2017)
experiment: S. Caldwell et al, arXiv:1706.06562 (2017)

Inspired by:
FM gate theory: Beaudoin et al PRA 86, 022305 (2012)
experiment: Strand et al, PRB 87, 220505(R) (2013)
B-tune gate: McKay et al Phys Rev Applied 6, 064007 (2016)

Rigetti 19Q

Rigetti	f_r MHz	f_q MHz	anharm MHz	T1 μs	T2 μs	F_{1q} %	F_{RO} %
0	5592	4386	-208	15	7	98.15	93.8
1	5703	4292	-210	18	8	99.07	95.8
2	5599	4221	-210	18	11	98.13	97.0
3	5708	3829	-224	31	17	99.08	88.6
4	5633	4372	-220	23	5	98.87	95.3
5	5178	3690	-224	22	11	96.45	96.5
6	5356	3809	-208	27	27	99.05	84.0
7	5164	3531	-216	29	13	99.16	92.5
8	5367	3707	-208	25	14	98.69	94.7
9	5201	3690	-214	21	11	99.34	92.7
10	5801	4595	-194	17	11	99.16	94.2
11	5511	4275	-204	17	5	99.01	90.0
12	5825	4600	-194	8	11	99.02	94.2
13	5523	4434	-196	19	13	99.33	92.1
14	5848	4552	-204	14	9	99.16	94.7
15	5093	3733	-230	21	7	98.52	97.0
16	5298	3854	-218	17	8	99.06	94.8
17	5097	3574	-226	24	8	98.95	92.1
18	5301	3877	-216	17	13	94.96	93.0
19	5108	3574	-228	25	10	99.42	93.0
	MHz	MHz	MHz	μs	μs	%	%
average	5445	4029	212	20	11	98.6	93.5
std	260	378	11	5	5	1.1	3.3

19Q connectivity g



it interactions

g between
cy and tunable
transmon qubits

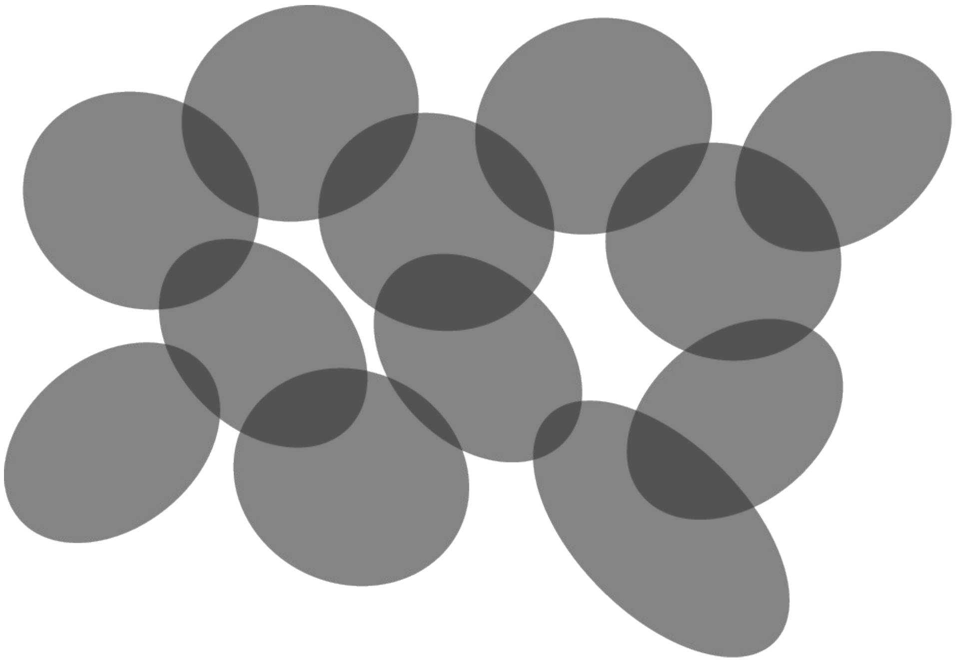
metric gates use RF
on to turn on
nance conditions

it gate fidelity of

Parametric gates:
theory: N. Didier et al, arXiv:1706.06566
experiment: S. Caldwell et al, arXiv:1706.

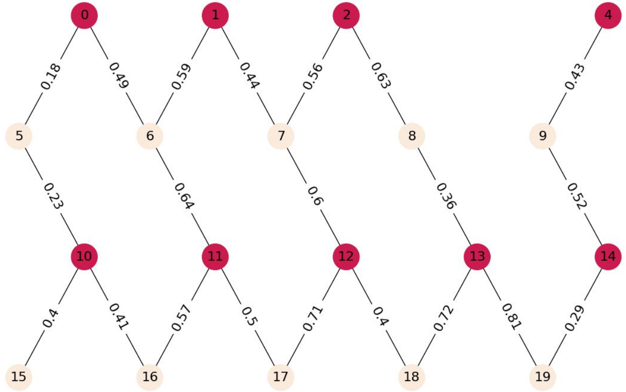
Inspired by:
et al PRA 86, 022305 (2012)
al, PRB 87, 220505(R) (2013)
B-tune gate: McKay et al Phys Rev Applied 6, 064007 (2016)

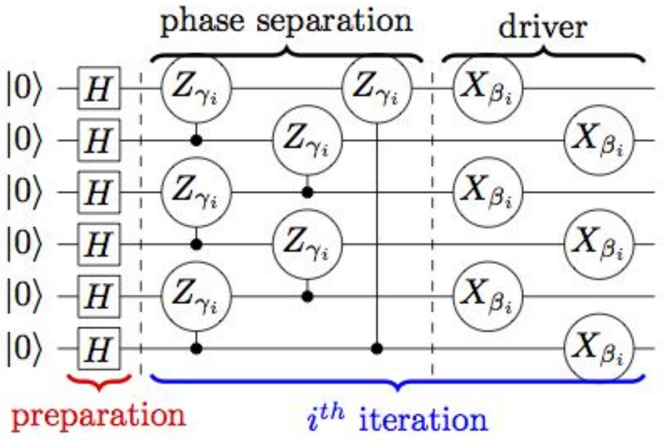
Sparse similarity metrics



Similarity is measured by a ***distance metric*** over the feature vector. In some domains, similarity can be measured by an ***overlap*** metric, leading to ***sparse*** graphs.

Sparse graphs with 19Q connectivity



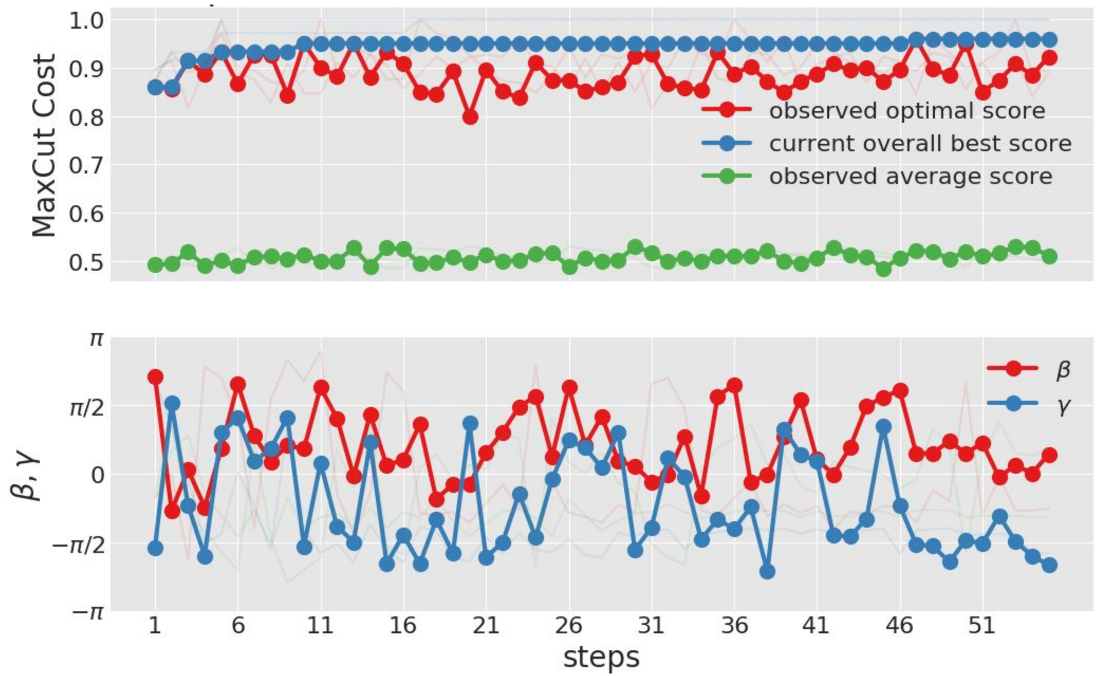


Generate a family of sparse graphs with ***random weights*** matching the connectivity of 19Q.

This allows implementation of H_C in a circuit of depth 3 (becomes depth 6 after compilation)

Clustering on 19Q

QAOA with 2,500 samples at each step

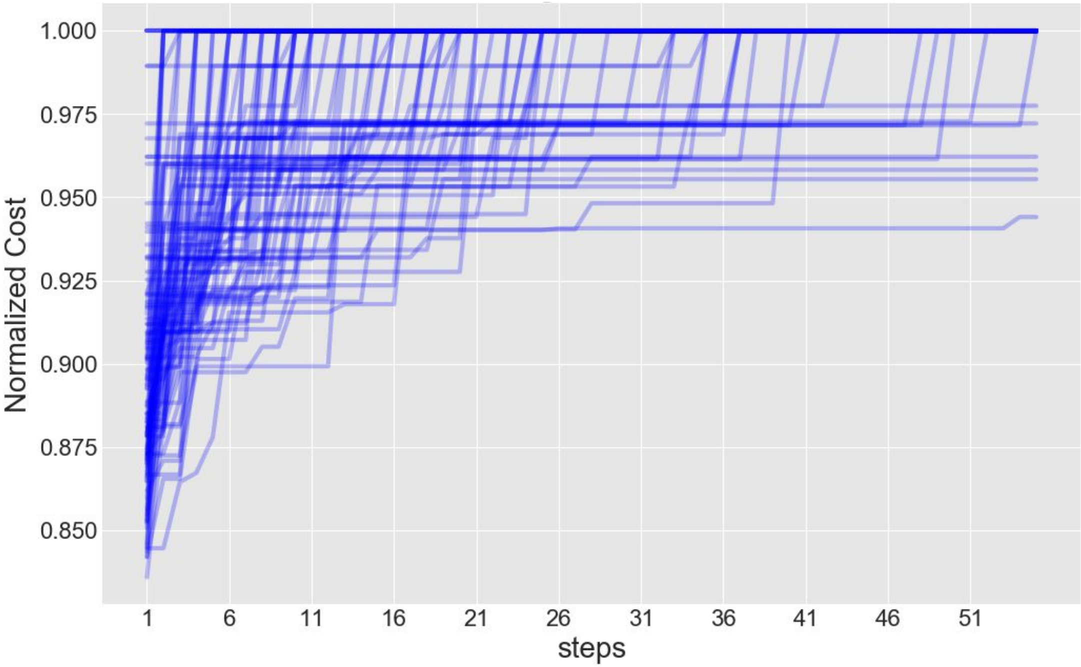


We run QAOA with p=1 on 19Q. The ***average cost*** is typically quite low, but we observe some samples close to the optimal solution.

We use a Bayesian approach to choose **β**, **γ**

Clustering on 19Q

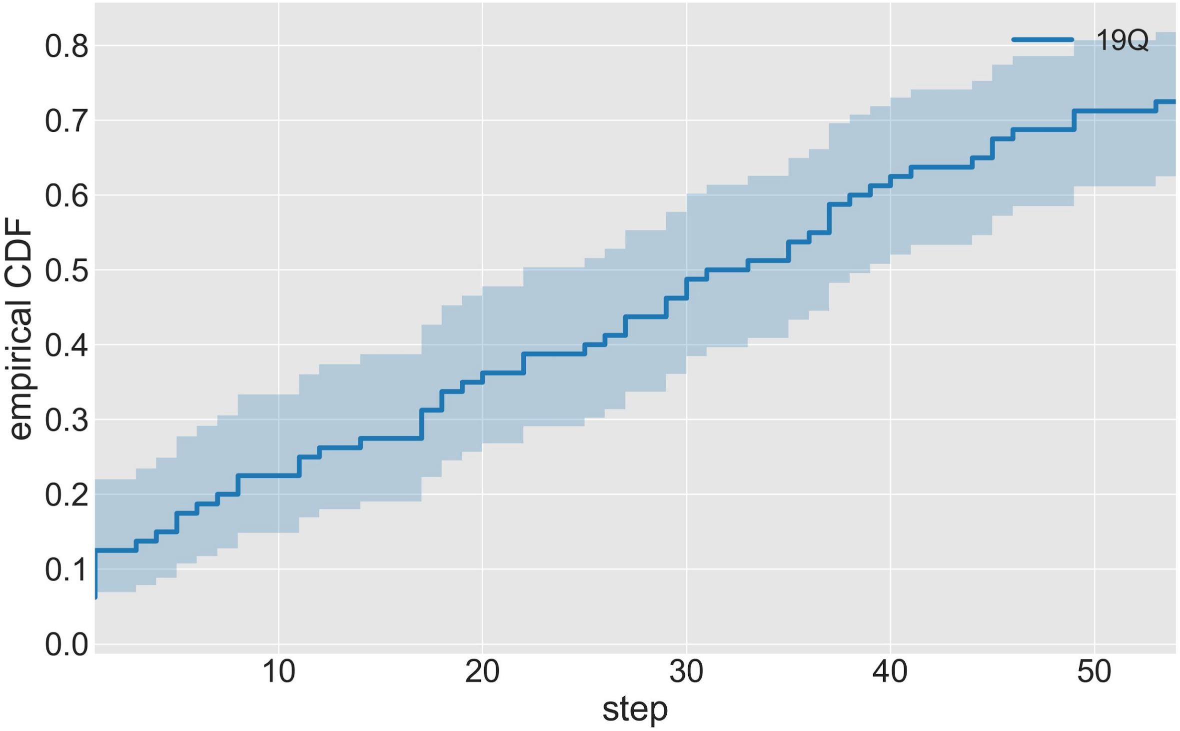
83 trials for a fixed problem instance



In many such trials, the algorithm actually finds the optimal solution.

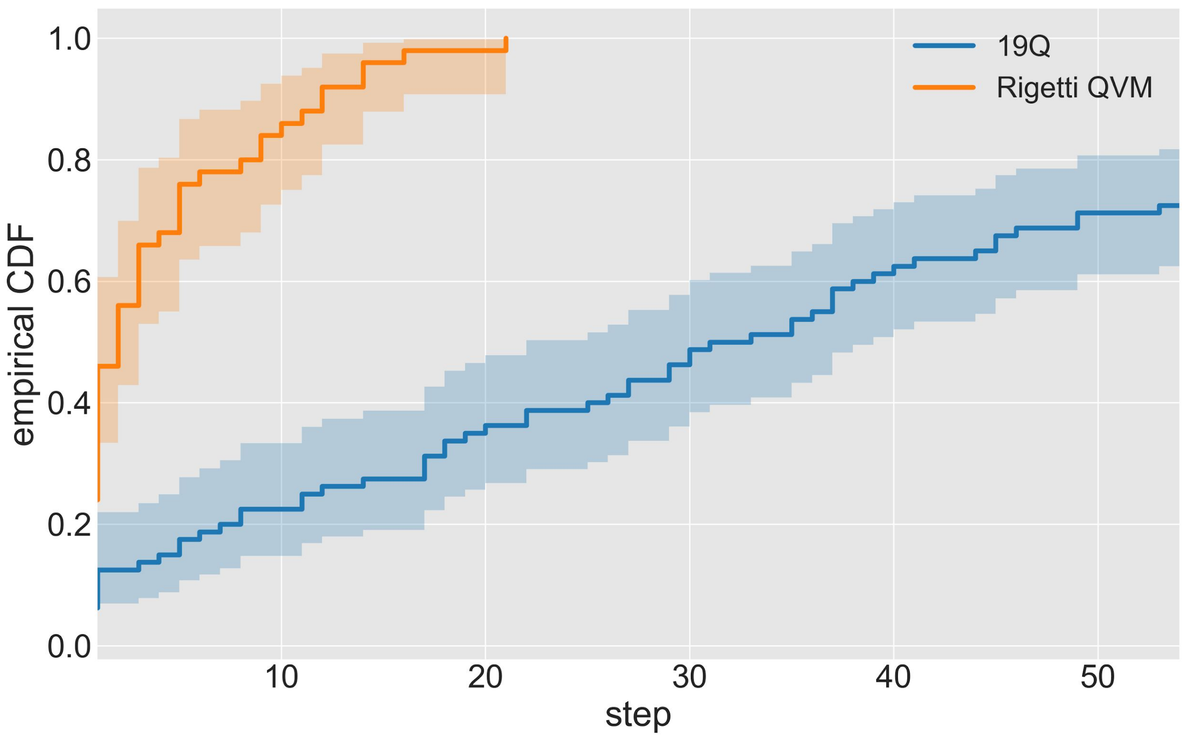
From these trials we calculate an empirical CDF.

Clustering performance



Success probability monotonically increases with number of steps.

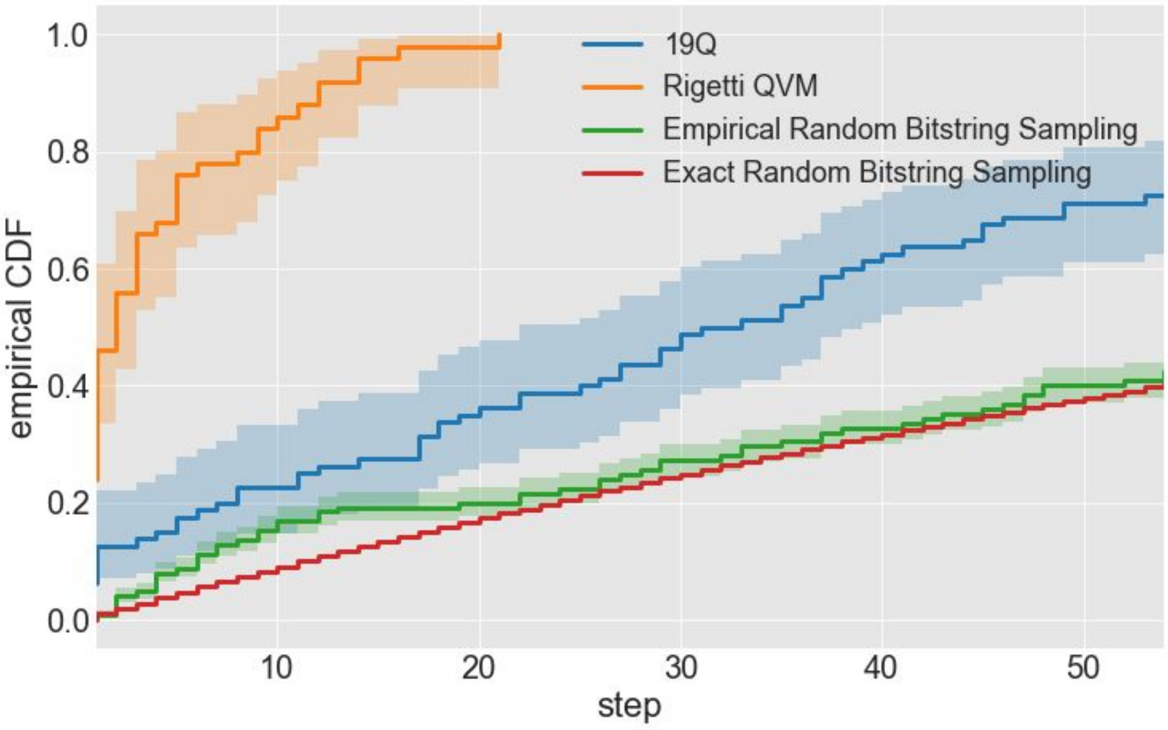
Clustering performance



Success probability monotonically increases with number of steps.

Noise in 19Q has a significant impact on performance.

Clustering performance



Success probability monotonically increases with number of steps.

Noise in 19Q has a significant impact on performance.

Approach clearly outperforms random sampling.

Otterbach, … WZ, ... et al 1712.05771 Unsupervised Machine Learning on a Hybrid Quantum Computer