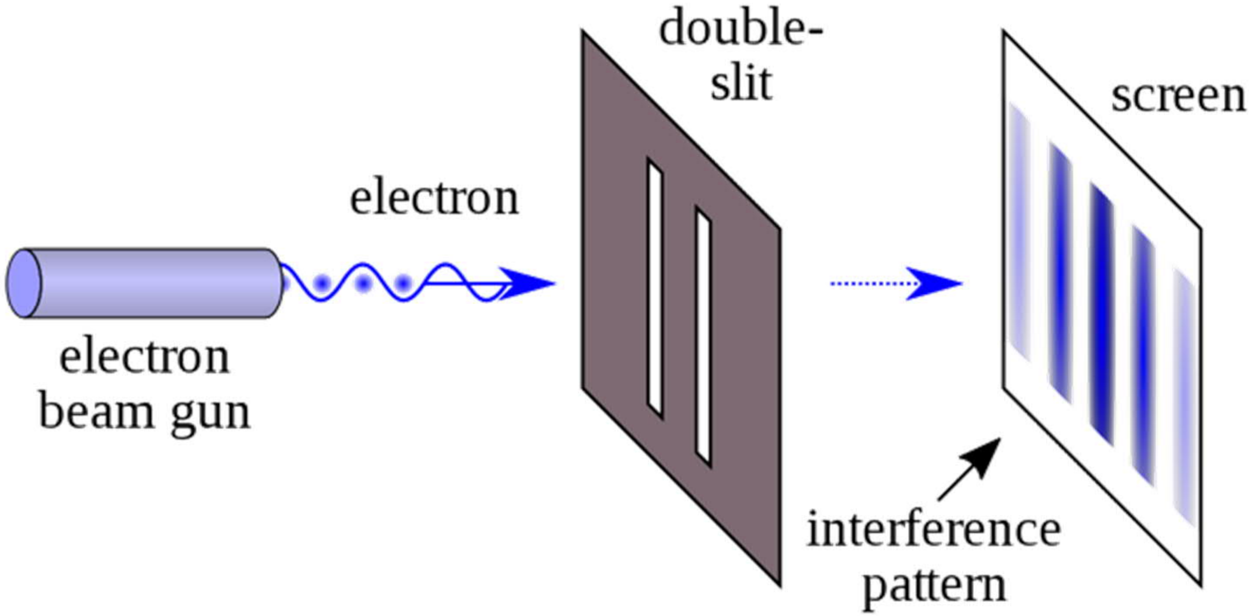
Towards quantum algorithms for real-world problems

Ville Kotovirta (& Hannu Reittu)
Quantum algorithms and software / VTT

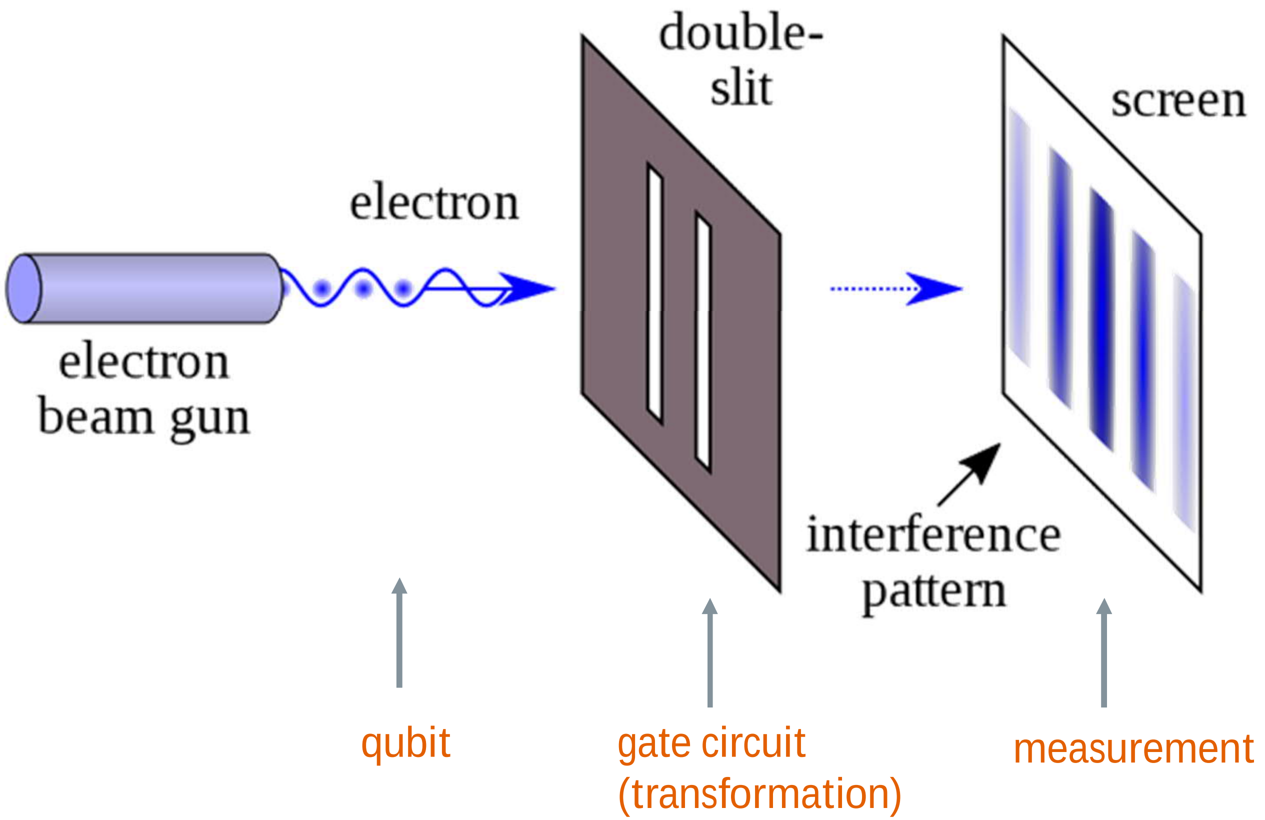
Contents

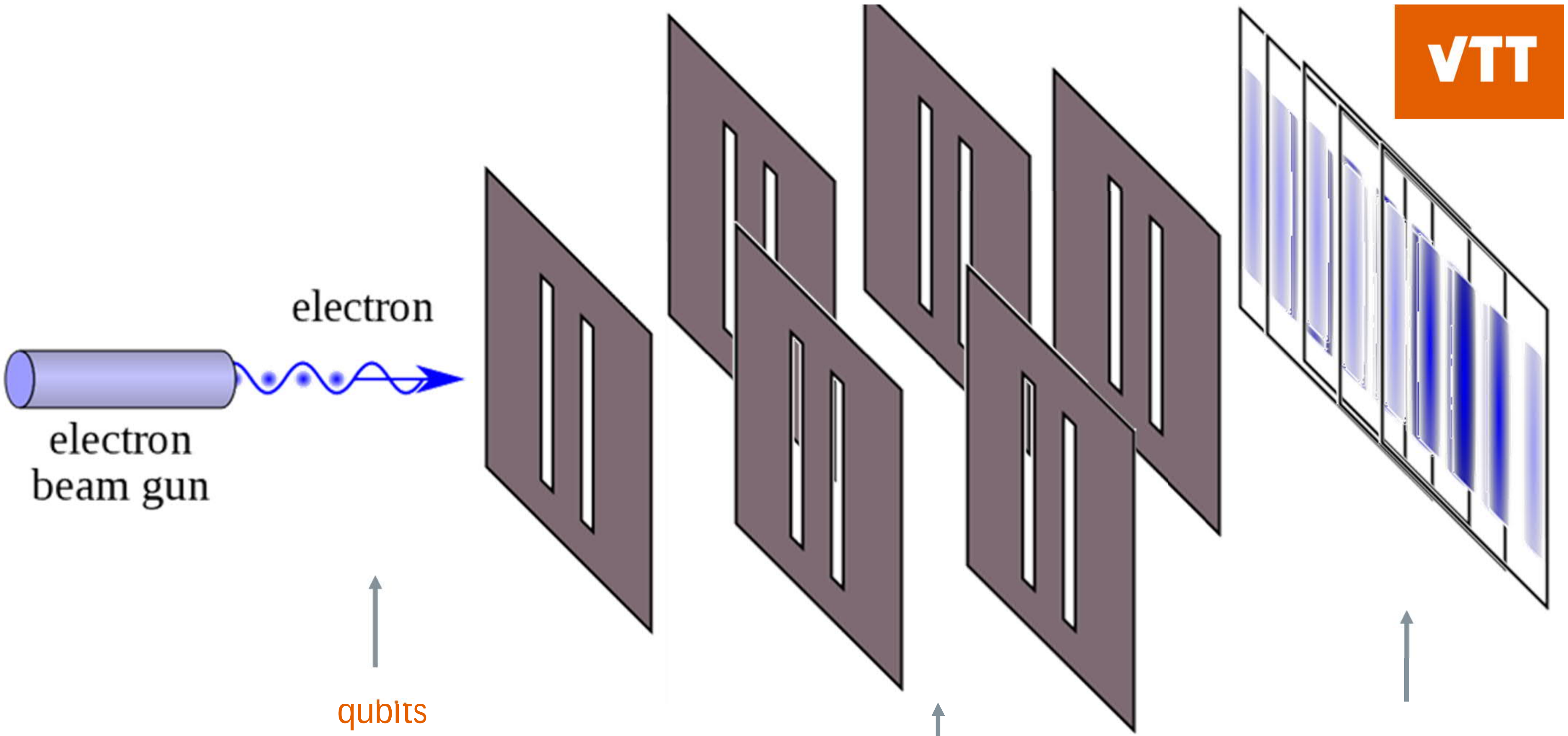
- Quantum computing?
- VTT aims
- A story of one problem
  - .. and algorithms to solve it based on quantum annealing and quantum gate computing
- What's next?



$$i\hbar\frac{\partial}{\partial t}|\psi(t)\rangle = \hat{H}|\psi(t)\rangle$$

Schrödinger equation

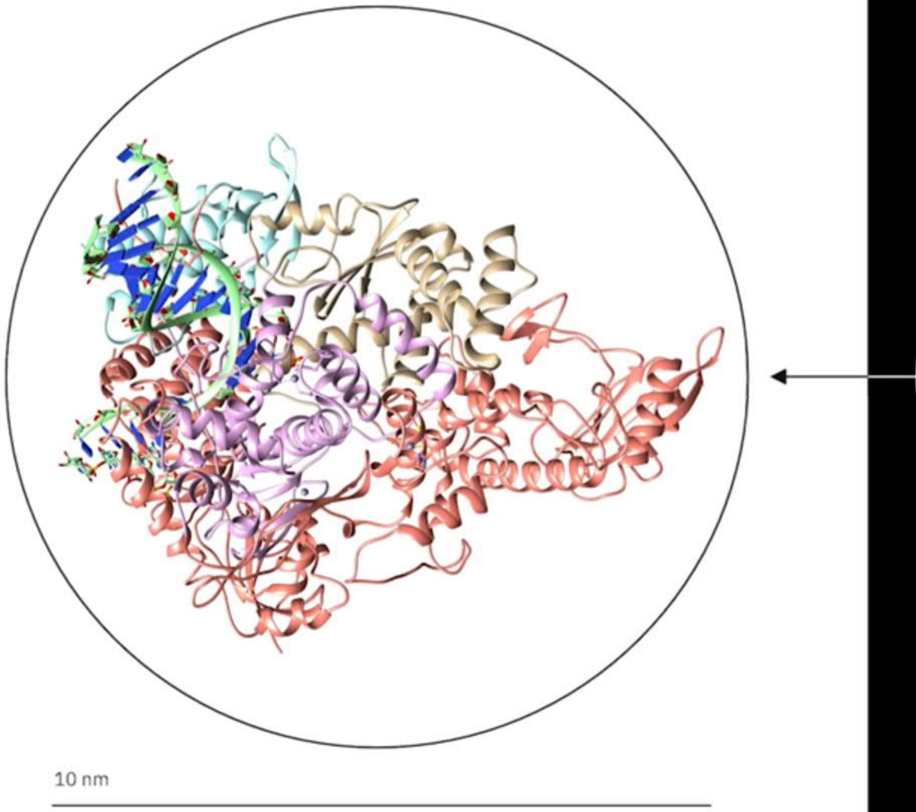

double-
slit
screen
electron
electron
beam gun
interference
pattern
qubit
gate circuit
(transformation)
measurement

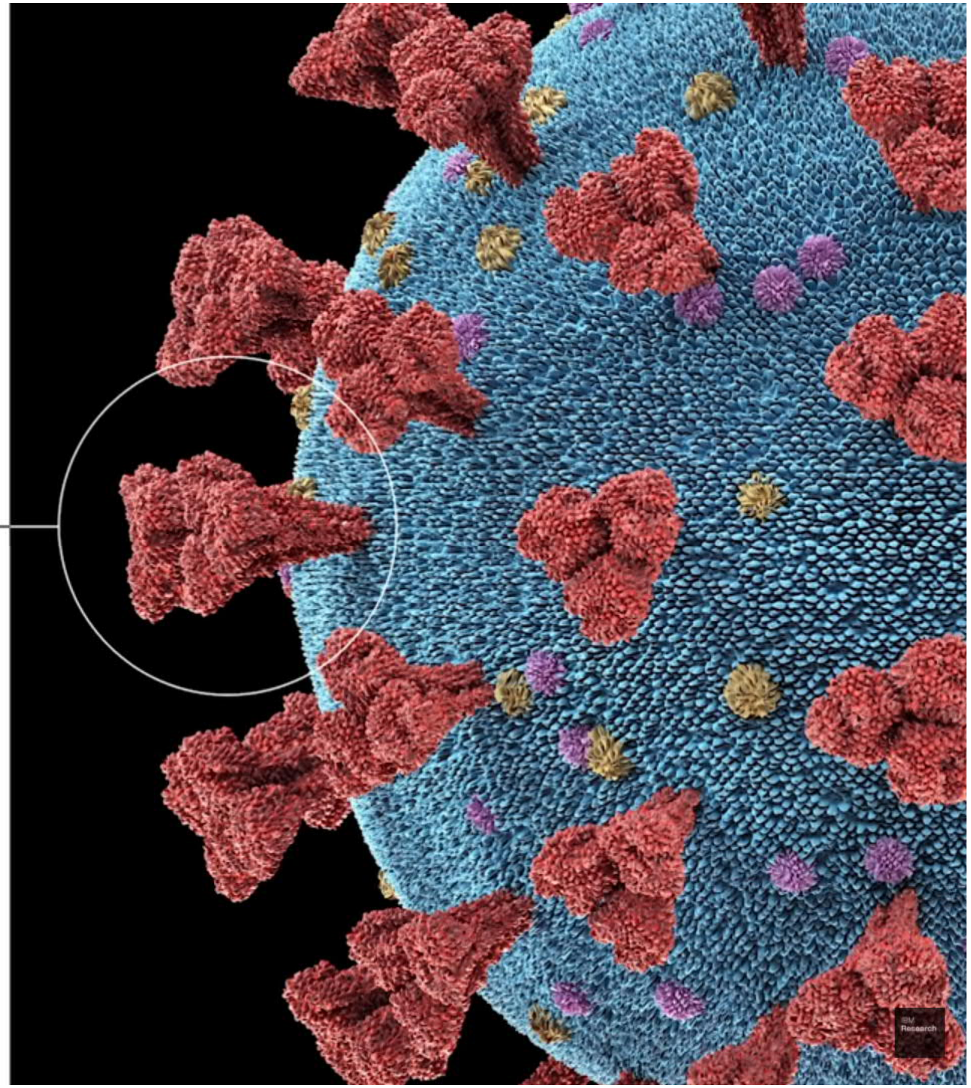


qubits

Programmable circuit

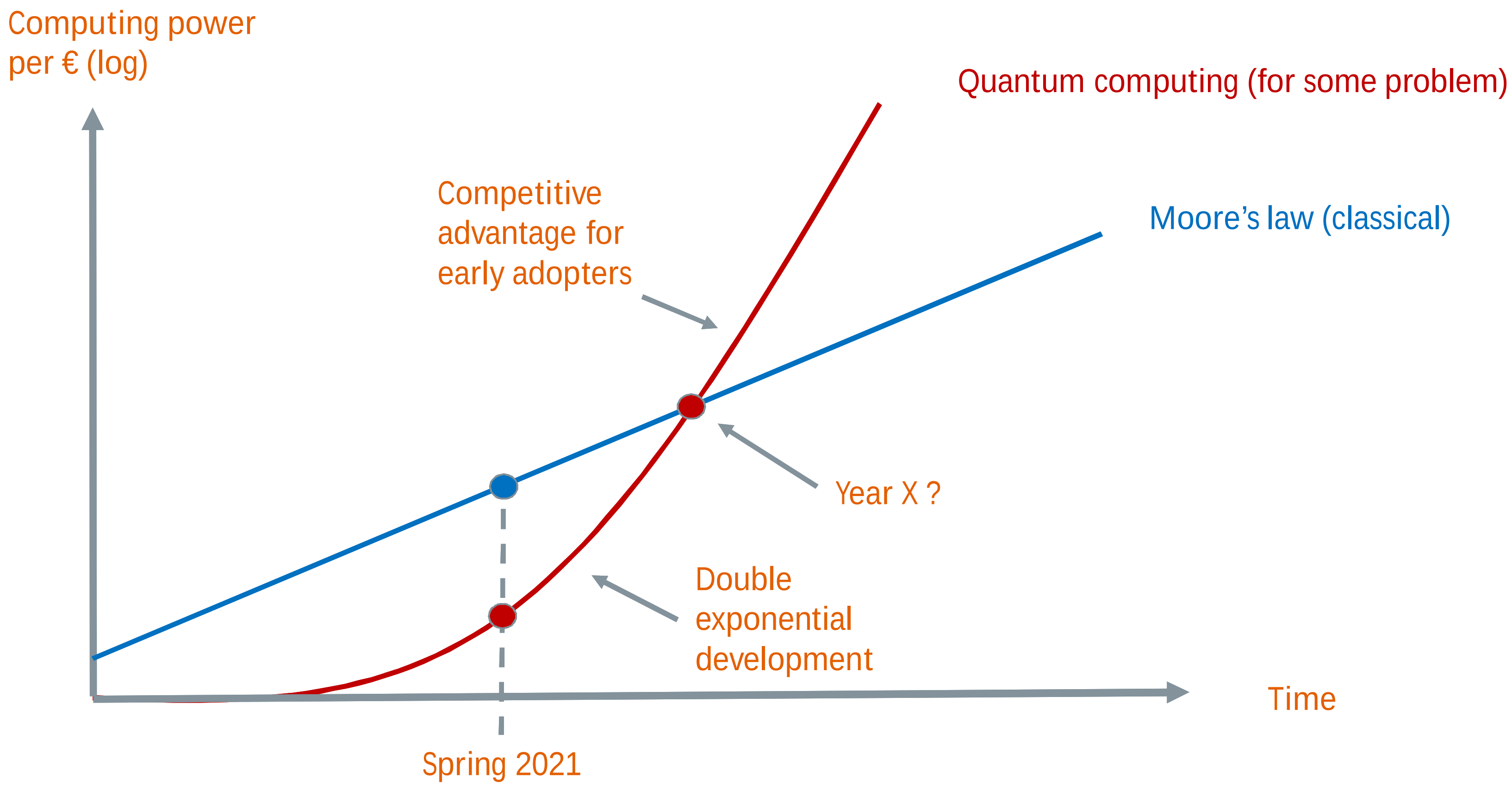
Measurement
= useful
computation



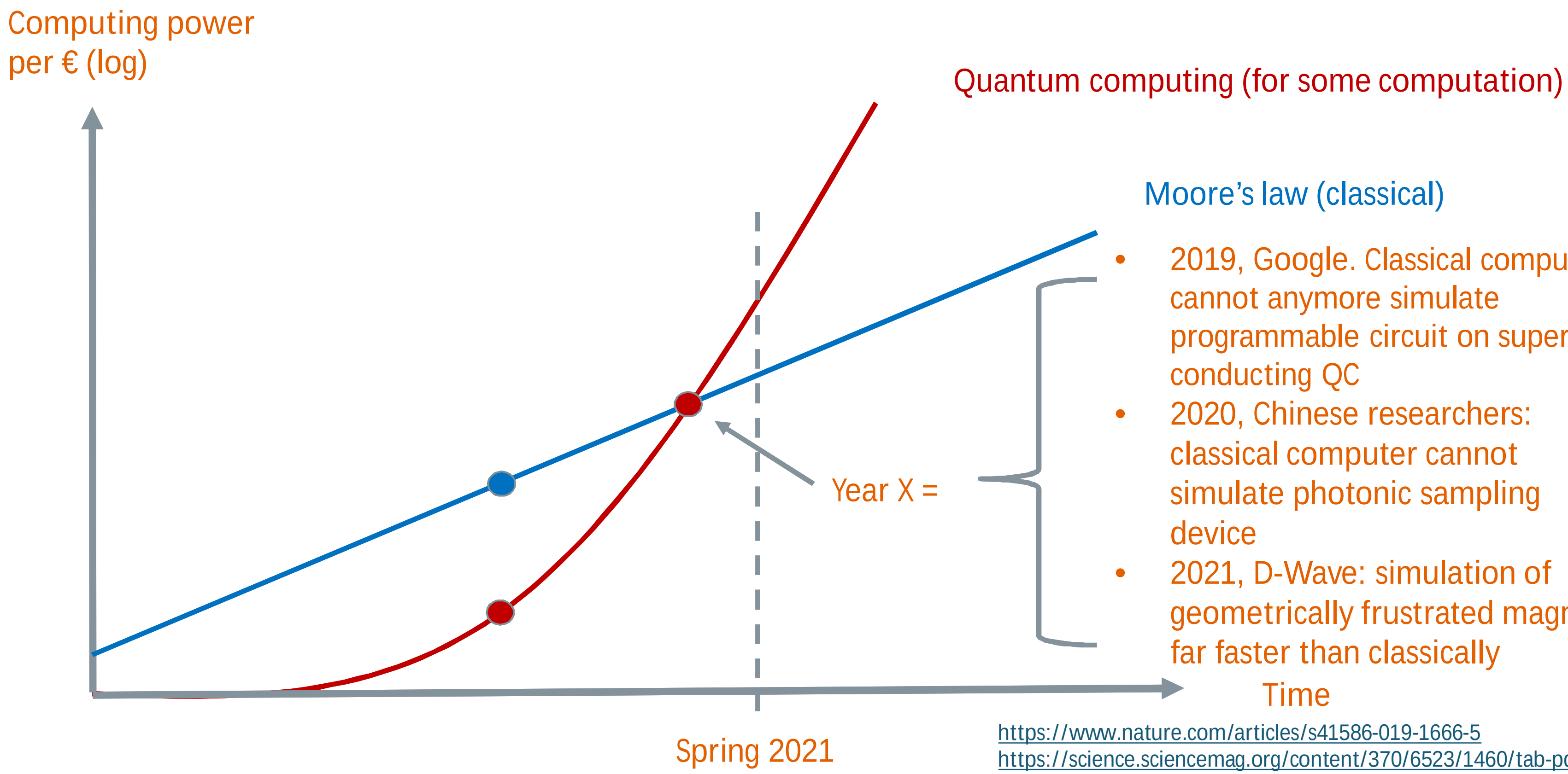


Source: The COVID-19 High Performance Computing Consortium
PI: Prof. Andrés Cisneros, University of North Texas

Dramatic shift is inevitable



Dramatic shift is inevitable

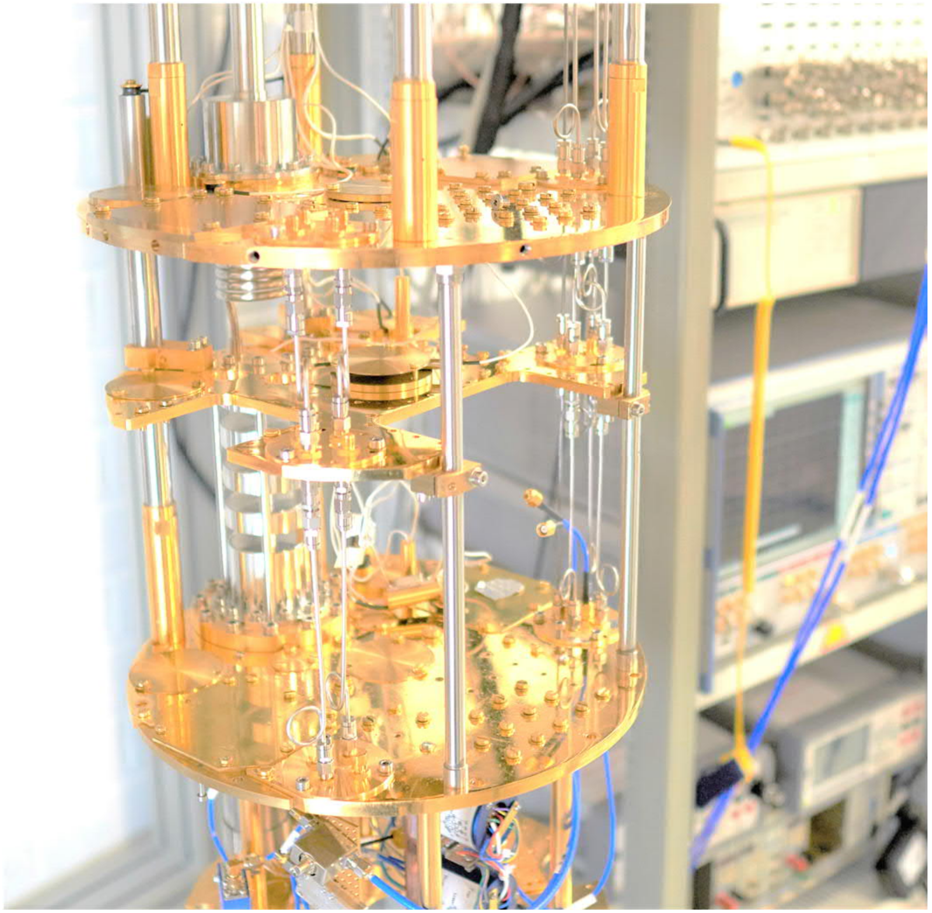


- 2019, Google. Classical computer cannot anymore simulate programmable circuit on super-conducting QC
- 2020, Chinese researchers: classical computer cannot simulate photonic sampling device
- 2021, D-Wave: simulation of geometrically frustrated magnets far faster than classically

https://www.nature.com/articles/s41586-019-1666-5
https://science.sciencemag.org/content/370/6523/1460/tab-pdf
https://www.nature.com/articles/s41467-021-20901-5

The Quantum computer build project at VTT

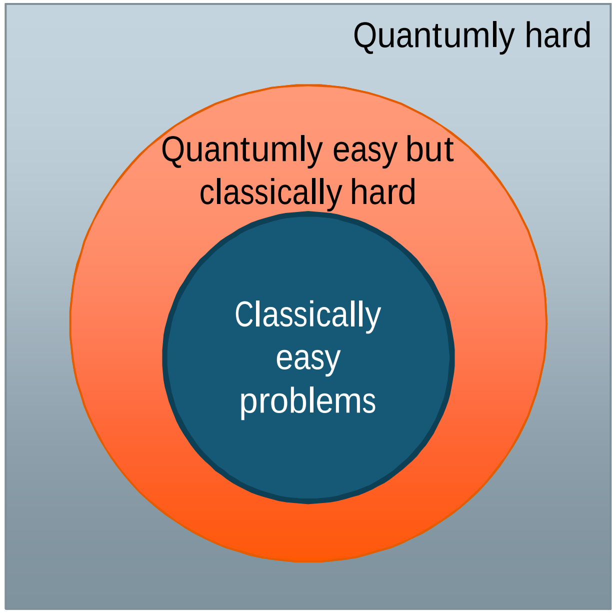
- Based on 20,7M€ funding received from Govt. of Finland
- Co-innovation project with Finnish start-up IQM resulting from a public procurement process
- Project runs from 2020 – 2024
- 3-phase project with targets to build at least 5, 20 and 50 qubit superconducting machines



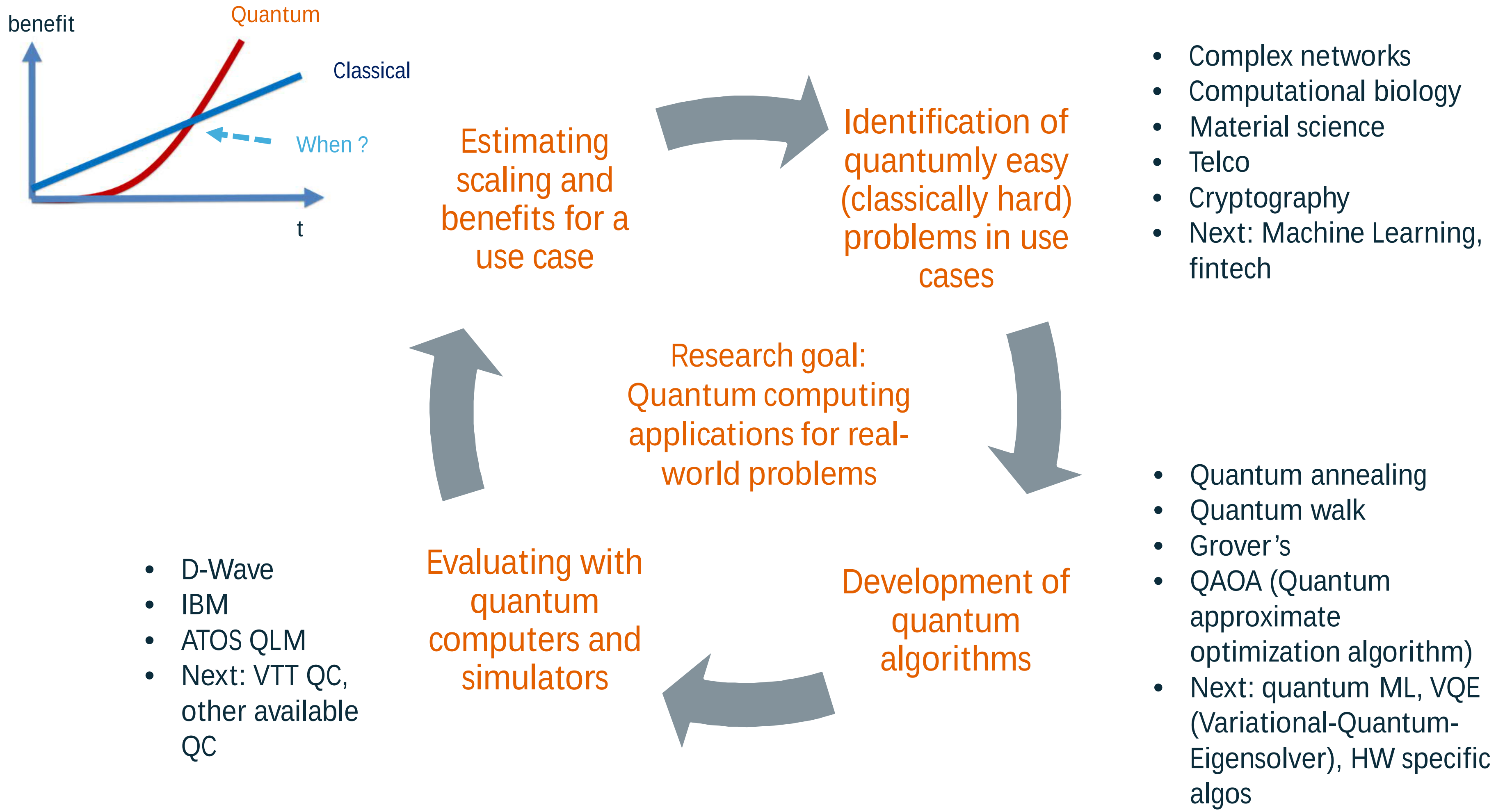
Main research questions

- What problems within selected business domains and use cases are hard for classical computing, but easy for quantum computing?
- Can we develop quantum algorithms to solve the problems?
- How do they scale?
- What kind of QC would be needed to run them more efficiently than classical algorithms?

Computation Problems



© John Preskill

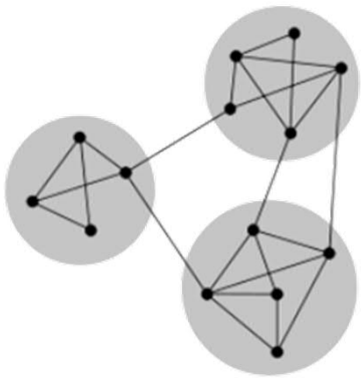

benefit
Quantum
Classical
When ?
t
Estimating scaling and benefits for a use case
Identification of quantumly easy (classically hard) problems in use cases
Complex networks
Computational biology
Material science
Telco
Cryptography
Next: Machine Learning, fintech
Research goal: Quantum computing applications for real-world problems
Quantum annealing
Quantum walk
Grover's
QAOA (Quantum approximate optimization algorithm)
Next: quantum ML, VQE (Variational-Quantum-Eigensolver), HW specific algos
Development of quantum algorithms
Evaluating with quantum computers and simulators
D-Wave
IBM
ATOS QLM
Next: VTT QC, other available QC

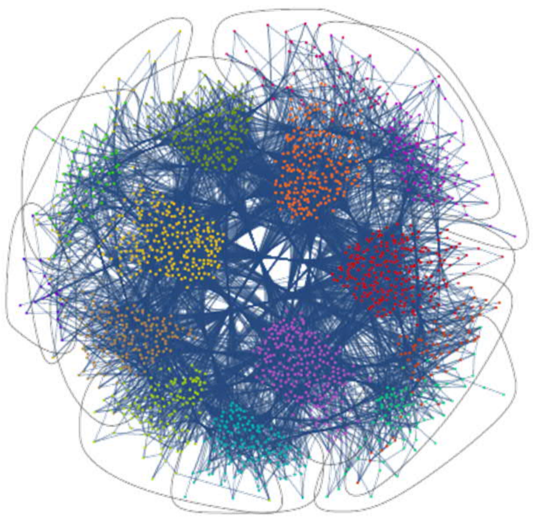
A story of one problem – development of Community panning *)

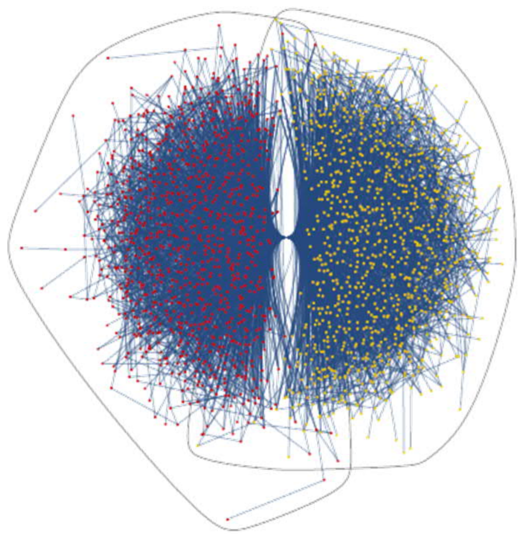
*) Hannu Reittu: theory and algorithm, Ville Kotovirta: programming and testing on QC and simulators

Complex networks

- Complex networks - a complex network is a graph (network) with non-trivial topological features - often occur in networks representing real systems, like computer networks, biological networks, technological networks, brain networks, climate networks, social networks …
- Problem: community detection







Szemerédi's Regularity Lemma

- Szemerédi's Regularity Lemma (1976) (SRL)
  - a major result in 'extremal graph theory'
  - huge number of other theoretical results come from SRL
  - SRL $\rightarrow$ Green and Tao: there exists arbitrarily long arithmetic series of prime numbers ($\mathbb{P}$): $\forall s \in \mathbb{N}, \exists p, a : p \in \mathbb{P}, a \in \mathbb{N}$ s.t. $p + ka \in \mathbb{P}, k = 0, 1, 2, \cdots, s$ $\rightarrow$ Fields Medal

Figure: Terry Tao wins Fields Medal in 2006

Szemerédi's Regularity Lemma in brief

- any large graph has a low complexity representation as a collection of bounded number of random like bipartite graphs (regular pairs)
- justifies a kind of stochastic block model
- can be found in poly-time (in theory)

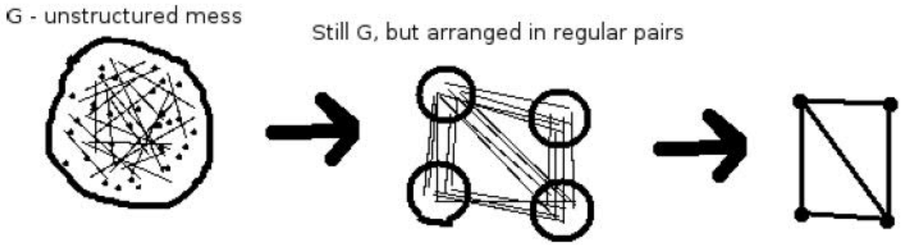


Figure: A caricature of SRL (Google Images)

Bipartite graph

- a bipartite graph with bipartition (V_1, V_2)

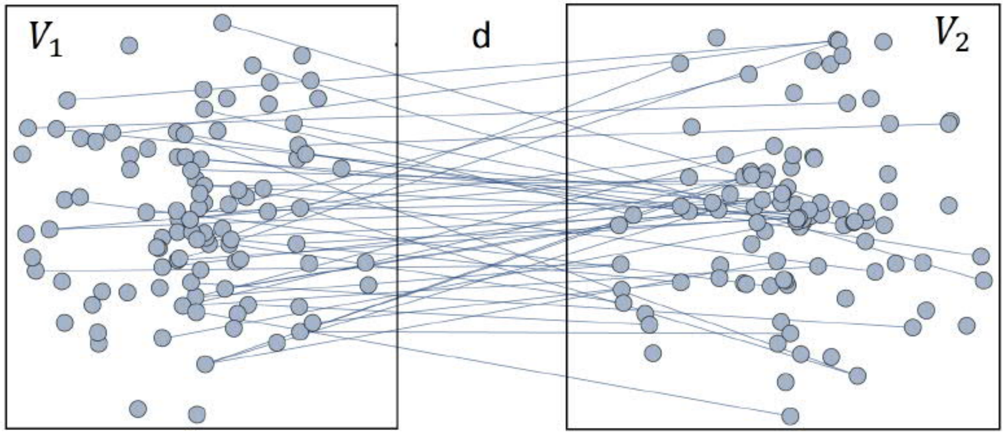


Figure: Links only between V_1 and V_2, link density d

ϵ-regular pair

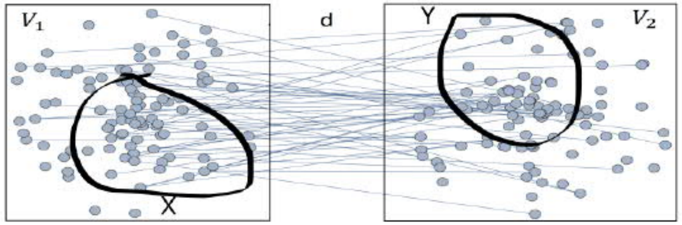


Figure: A regular pair? Check for all subsets X and Y

- a bipartite graph is called ϵ-regular (by T.Tao)
- iff for all subsets $X \subset V_1, Y \subset V_2$:

$$|X||Y|(d(V_1, V_2) - d(X, Y)) = O(\epsilon|V_1||V_2|)$$

$$d(X, Y) = \frac{e(X, Y)}{|X||Y|}, \quad d(V_1, V_2) = \frac{e(V_1, V_2)}{|V_1||V_2|}$$

and where $e(S, M)$ is number of links joining S and M; $|M|$ is number of elements in a set M.

Tao's function for regularity check

Cost function for regularity check:

$$L(X, Y) := |X||Y|d(V_1, V_2) - e(X, Y) =$$
$$\mathbb{E}\, e(X, Y) - e(X, Y)$$

Range of L defines level of regularity:

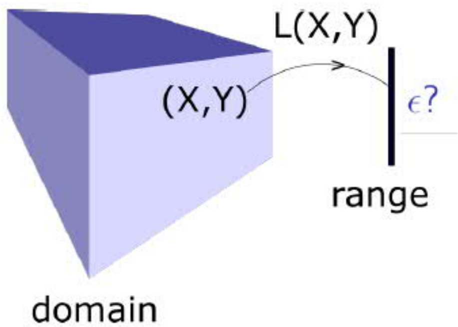


Figure: Function L has huge domain. Its range defines ϵ. Range is hard to find!

Regularity check is probably a quantumly hard problem !

$\min L(X, Y)$?

$$L(X, Y) := \mathbb{E}\, e(X, Y) - e(X, Y).$$

what is meaning of

$$\min_{X,Y} L(X, Y)?$$

Answer: finding maximally large and dense subgraph

Community detection: main idea *)

$$(X^*, Y^*) = \underset{(X,Y)}{\arg\min}\, L(X, Y) := \underset{(X,Y)}{\arg\min}(|X||Y|d(V_1, V_2) - e(X, Y)).$$

- corresponds in finding subsets where $e(X^*, Y^*) >> |X^*||Y^*|d(V_1, V_2)$
- in other words, induced subgraph (X^*, Y^*) is better connected than the whole graph in average
- communities are subgraphs that have large internal connectivity compared with connectivity to other communities
- finding communities is related to finding dense subgraphs

*) Reittu, Kotovirta, et al. Towards analyzing large graphs with quantum annealing. https://ieeexplore.ieee.org/document/9006174

Community detection algorithm

Reduction to min $L(X, Y)$ and bipartization

- at the top: a graph with unknown communities
- flip a coin to split nodes into two sets (left and right =bipartition)
- only links between left - and right parts are preserved.
- every community is split into two parts

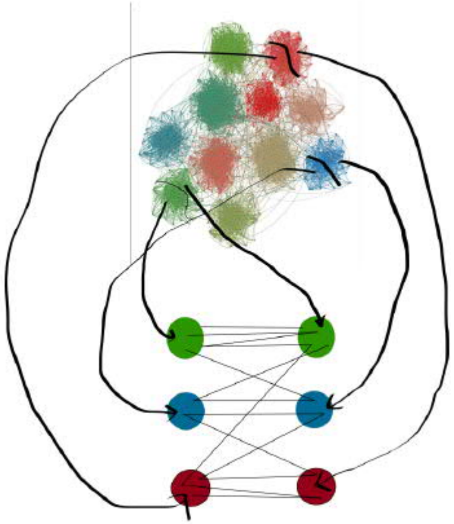
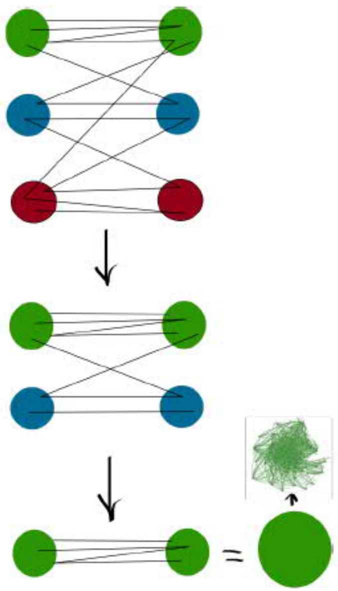
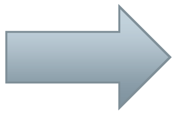


Figure: The firs step in community detection: forming a bipartite graph

Finding a community step by step

- apply min $L(X, Y)$ to a graph and take a graph induced by $(X^*, Y^*) := \arg\min_{(X,Y)\subset(V_1,V_2)} L(X, Y)$ as a new input...
- one round of the algorithm finds one community (green ball)
- by repeating all communities are found
- no need to know beforehand number of communities





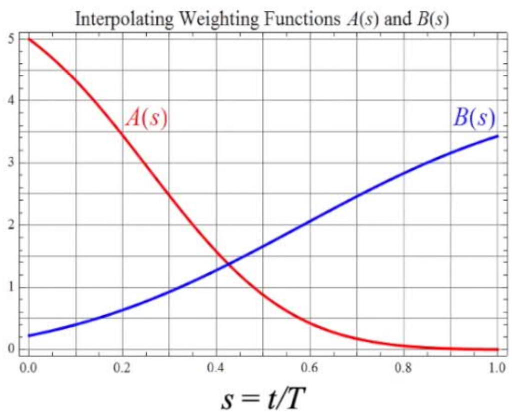
Optimisation problem

Quantum annealing for optimisation problems

Quantum Hamiltonian is an operator on Hilbert space:

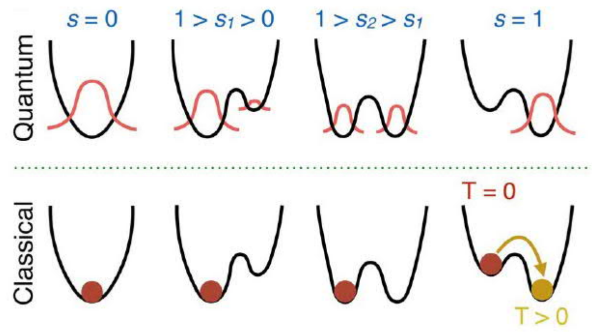
$$\mathcal{H}(s) = A(s)\sum_i \sigma_i^x + B(s)\left[\sum_i a_i\,\sigma_i^z + \sum_{i<j} b_{ij}\,\sigma_i^z\sigma_j^z\right]$$

transverse field



Corresponding classical optimization problem:

$$\mathrm{Obj}(a_i, b_{ij}; q_i) = \sum_i a_i q_i + \sum_{i<j} b_{ij} q_i q_j$$



Ref. D-wave: Quantum Computing Tutorial Part 1: Quantum annealing, QUBOs and more
https://www.youtube.com/watch?v=teraaPiaG8s&list=PLPvKnT7dgEsujrrP7i_mgbkivBN3J6A8u&index=1

Ising model

Starting point: $L(X,Y) := |X||Y|d(V_1,V_2) - e(X,Y) = \mathbb{E}\,e(X,Y) - e(X,Y)$

VTT

- assume a bipartite graph $G(V_1, V_2)$ with adjacency matrix A ($(A)_{i,j} := a_{i,j} = 1$ if there is a link between nodes i and j, otherwise $a_{i,j} = 0$)
- to each node $i \in V_1 \cup V_2$ assign a binary variable $s_i \in \{0,1\}$
- by definition $X = \{i \in V_1 : s_i = 1\}$ and $Y = \{i \in V_2 : s_i = 1\}$

As a result $|X||Y| = \sum_{i\in V_1, j\in V_2} s_i s_j$ and $e(X,Y) = \sum_{i\in V_1, j\in V_2} a_{i,j} s_i s_j$ and

$$L(X,Y) = \sum_{i\in V_1, j\in V_2} (d(V_1,V_2) - a_{i,j}) s_i s_j$$

Schematic view of Ising model

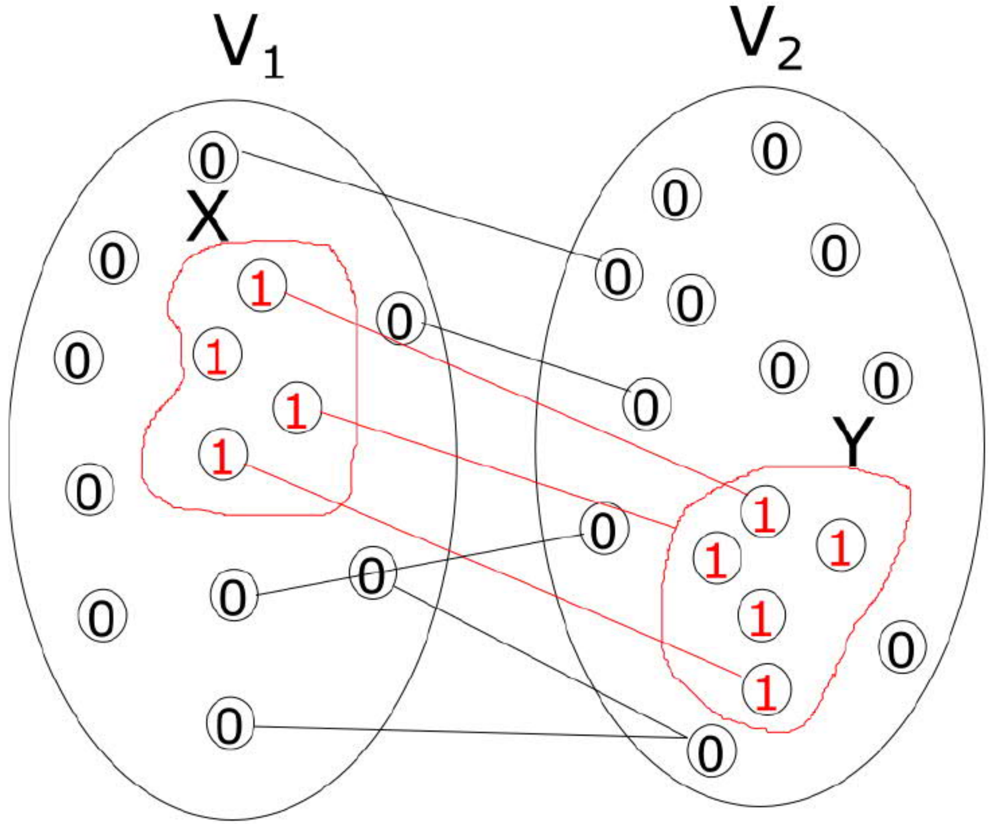


Figure: Binary variables with values 1 correspond to subsets X and Y in $L(X, Y)$)

Test it on D-wave

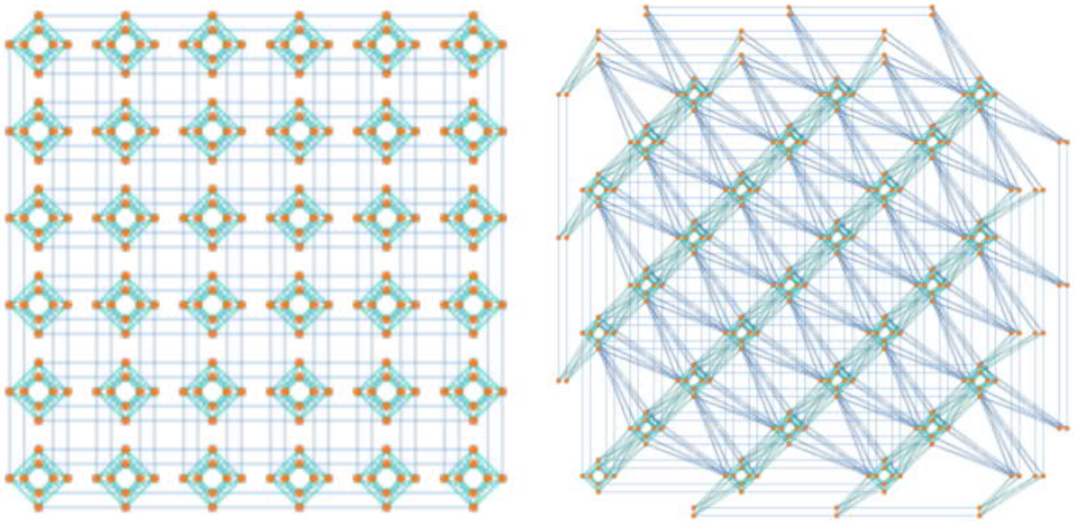


Figure 1: A C6 Chimera graph (left) with 36 unit cells containing 288 qubits. A P4 Pegasus graph (right) with 27 unit cells and several partial cells, containing 264 qubits. The comparatively rich connectivity structure of the P4 is clearly seen.

	2000Q	Advantage
Graph topology	Chimera	Pegasus
Graph size	C16	P16
Number of qubits	> 2000	> 5000
Number of couplers	> 6000	$> 35,000$
Couplers per qubit	6	15

Table 1: Typical characteristics of Chimera- and Advantage-generation QPUs.

Ref. D-Wave:The D-Wave Advantage System: An Overview
https://www.dwavesys.com/sites/default/files/14-1049A-A_The_D-Wave_Advantage_System_An_Overview.pdf

Results

Graph size	Simulated annealing	Q2000, 1000 runs	Q2000, 2000 runs	Advantage, 3000 runs	Advantage, 5000 runs	D-wave hybrid solver
50	-83.4084	-82.8032	-83.3472	-83.4084		
100	-231.8772			-212.4072	-214.7441	-231.8770
200	-678.0170					-678.0166
500	**-2605.45**18					**-2605.46**01
1000	**-9390.14**55					**-9390.20**10

D-wave hybrid found better results than laptop PC (qbsolver + simulated annealing) !

Quantum gate computing

- QAOA (Quantum Approximate Optimisation Algorithm) is a quantum gate algorithm to approximate the ground state of a k-local Hamiltonian (Farhi et al. 2014).
- QAOA can be used to approximate the ground state of an Ising model !
- For this we present our Hamiltonian as:

$$\mathcal{H}\left|x,y\right\rangle = \sum_{i,j} x_i w_{i,j} y_j \left|x,y\right\rangle$$

$$L(X,Y) = \sum_{i\in V_1, j\in V_2} (d(V_1,V_2) - a_{i,j}) s_i s_j$$

The algorithmic steps of the QAOA read as follows:

1. Generate the initial state as a uniform superposition of all states in the computational basis: $|\psi\rangle_i = H^{\otimes n} |0\rangle^{\otimes n}$.
2. Construct the unitary operator $U(\hat{H}, \gamma)$ which depends on the angle γ as follows:

$$U(\hat{H}, \gamma) = e^{-i\gamma\hat{H}} = \prod_{\alpha=1}^{m} e^{-i\gamma H_\alpha}. \tag{21}$$

3. Construct the operator B which is the sum of all single-bit σ^x operators:

$$B = \sum_{j=1}^{n} \sigma_j^x. \tag{22}$$

4. Define the angle-dependent quantum state for any integer $p \geq 1$ and $2p$ angles $\gamma_1 \dots \gamma_p \equiv \boldsymbol{\gamma}$ and $\beta_1 \dots \beta_p \equiv \boldsymbol{\beta}$ as follows:

$$|\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle = U(B, \beta_p)U(\hat{H}, \gamma_p) \dots U(B, \beta_1)U(\hat{H}, \gamma_1) |\psi_0\rangle. \tag{23}$$

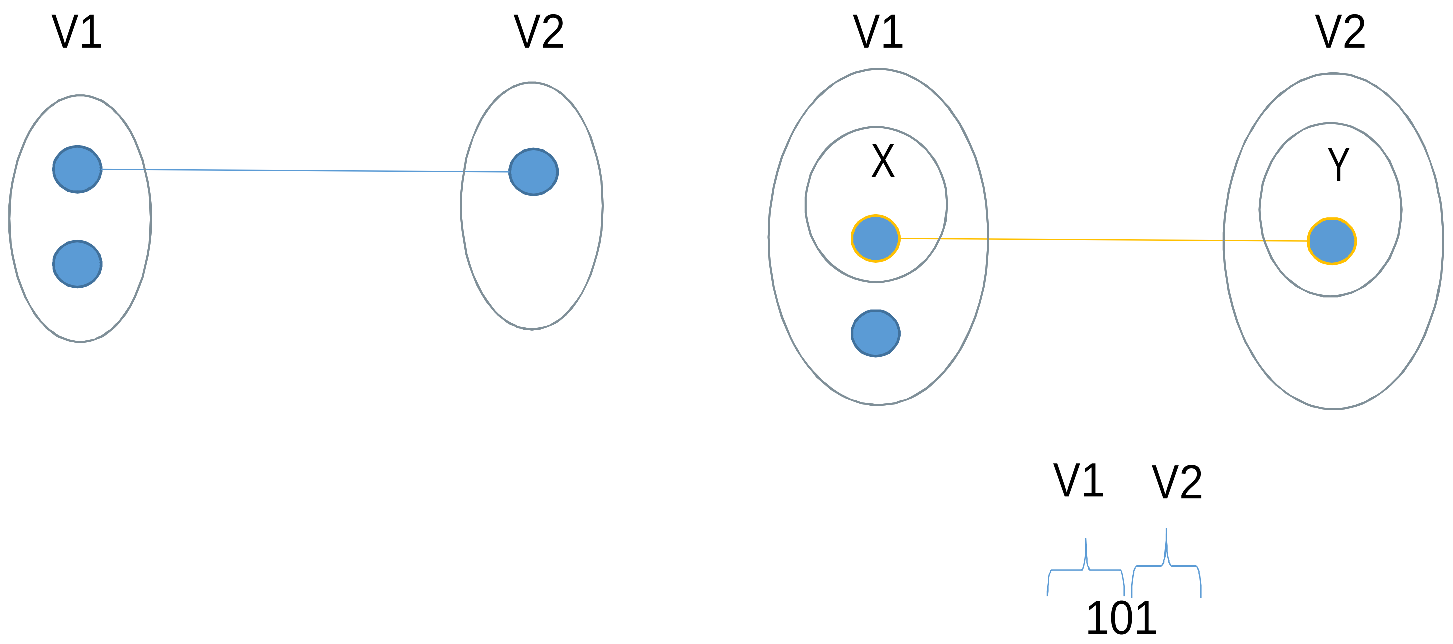
5. Obtain the expectation of $\hat{H}$ in this state (this step could be performed on a quantum computer),

$$F_p(\boldsymbol{\gamma}, \boldsymbol{\beta}) = \langle \boldsymbol{\gamma}, \boldsymbol{\beta}| \hat{H} |\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle. \tag{24}$$

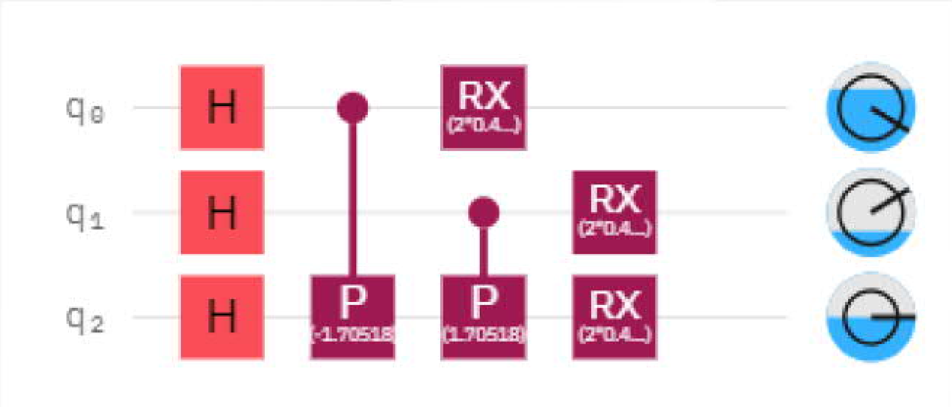
6. Update the parameters $(\boldsymbol{\gamma}, \boldsymbol{\beta})$ using a classical (or quantum) optimization algorithm in order to minimize F_p.
7. Iterate over steps 5 and 6 in order to find the minimum value of F_p for the near-optimal values $(\boldsymbol{\gamma}^*, \boldsymbol{\beta}^*)$.
8. Plug $(\boldsymbol{\gamma}^*, \boldsymbol{\beta}^*)$ into Equation (23) and evolve the initial state of the system to the state $|\boldsymbol{\gamma}^*, \boldsymbol{\beta}^*\rangle$.
9. Repeat step 8 with the same angles. A sufficient number of repetitions will produce a state which represents a close enough solution to the ground state of $\hat{H}$.

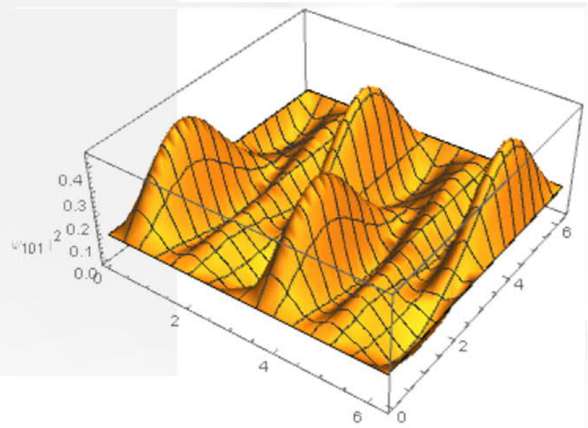
VTT

Zahedinejad & Zaribafiyan, 2017. Combinatorial Optimization on Gate Model Quantum Computers: A Survey https://arxiv.org/pdf/1708.05294.pdf

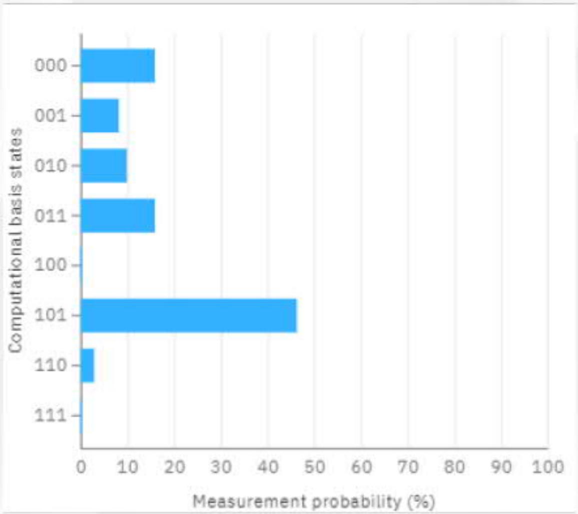

V1
V2
V1
V2
X
Y
V1
V2
101

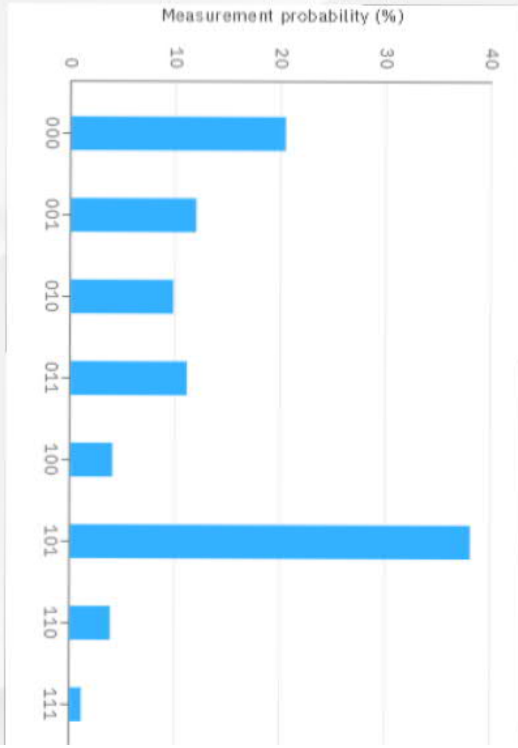
Optimal times: $t_1 = 0.499267$ and $\tau_1 = 1.70518$ with probability of the state $|q_2, q_1, q_0\rangle = |101\rangle$ equal to 0.463045.



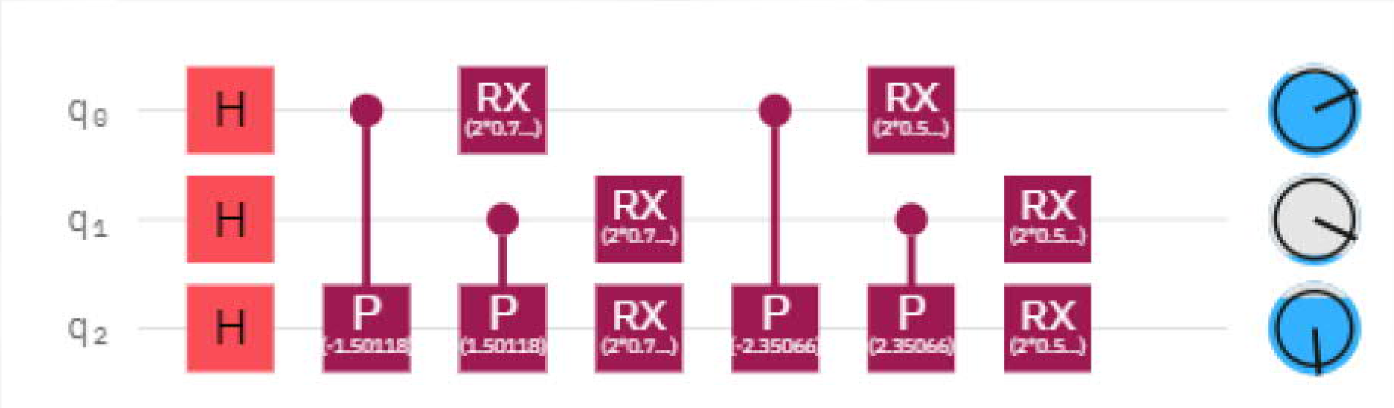


Simulations: $\mathbb{P}(|101\rangle) \approx 0.46$ and frequency of the same state among 8 thousand repetitions, on IBM's quantum computer Lima, is around 0.38.

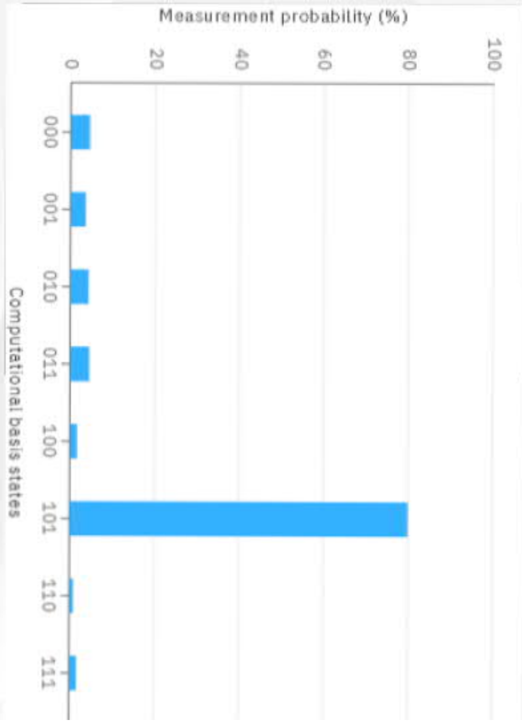


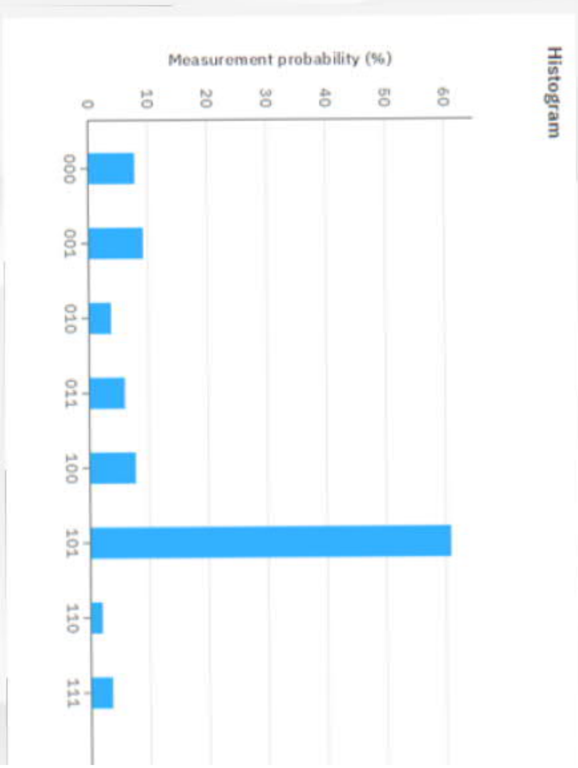


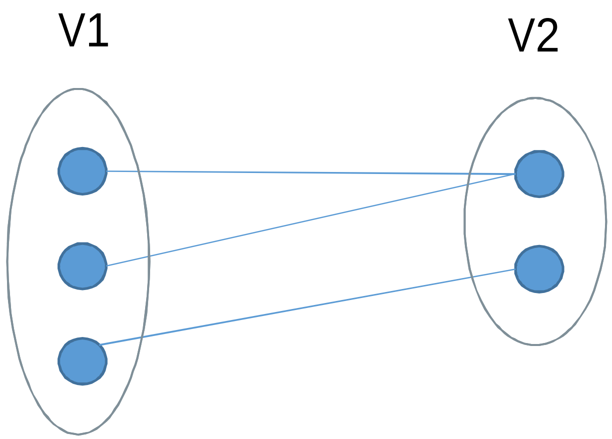
Next we made experiments with the graph and with QAOA having depth $p = 2$. We have now four parameters that can be optimized for increasing probability of the state $|101\rangle$. The found circuit has the corresponding probability around 0.8. The circuit with the optimal parameters is:

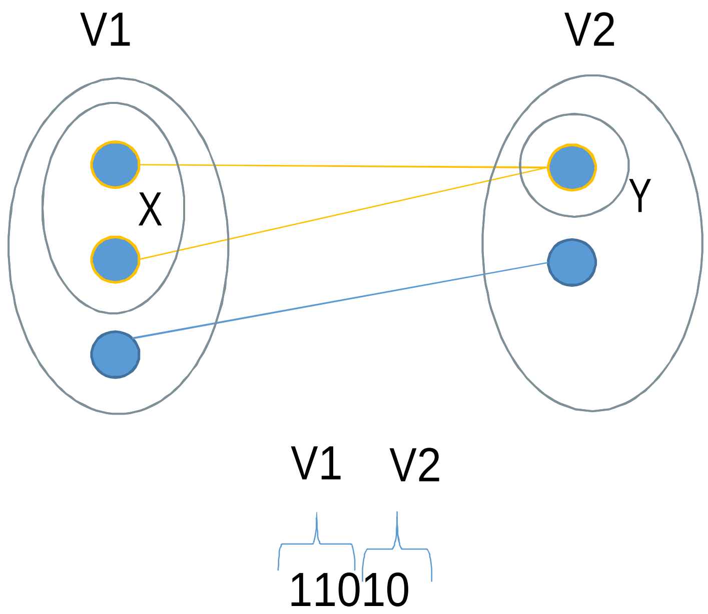


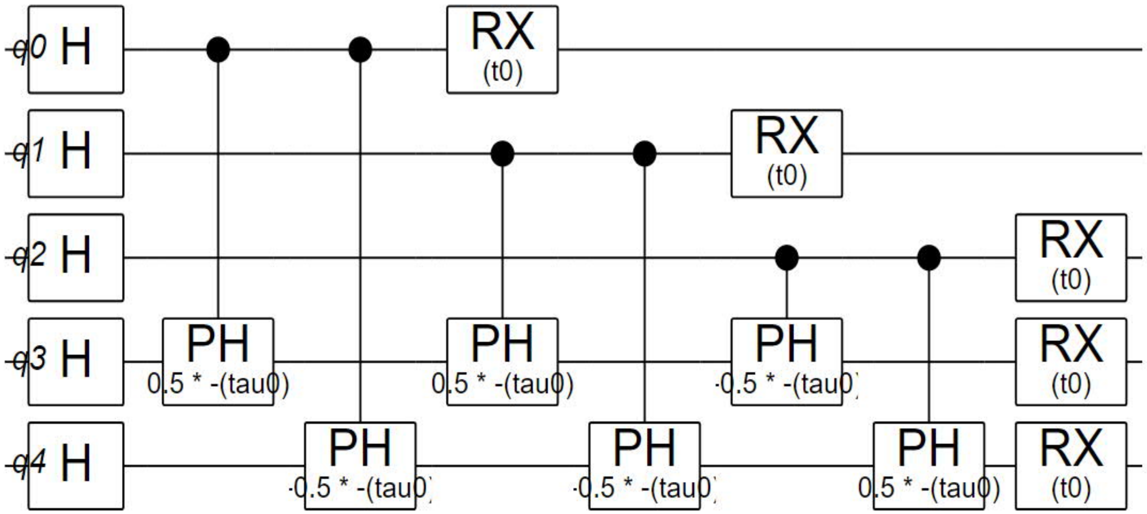
Simulated probability of $|101\rangle$ is 0.8 and the corresponding frequence of this state on IBM (Lima) machine is around 0.6.



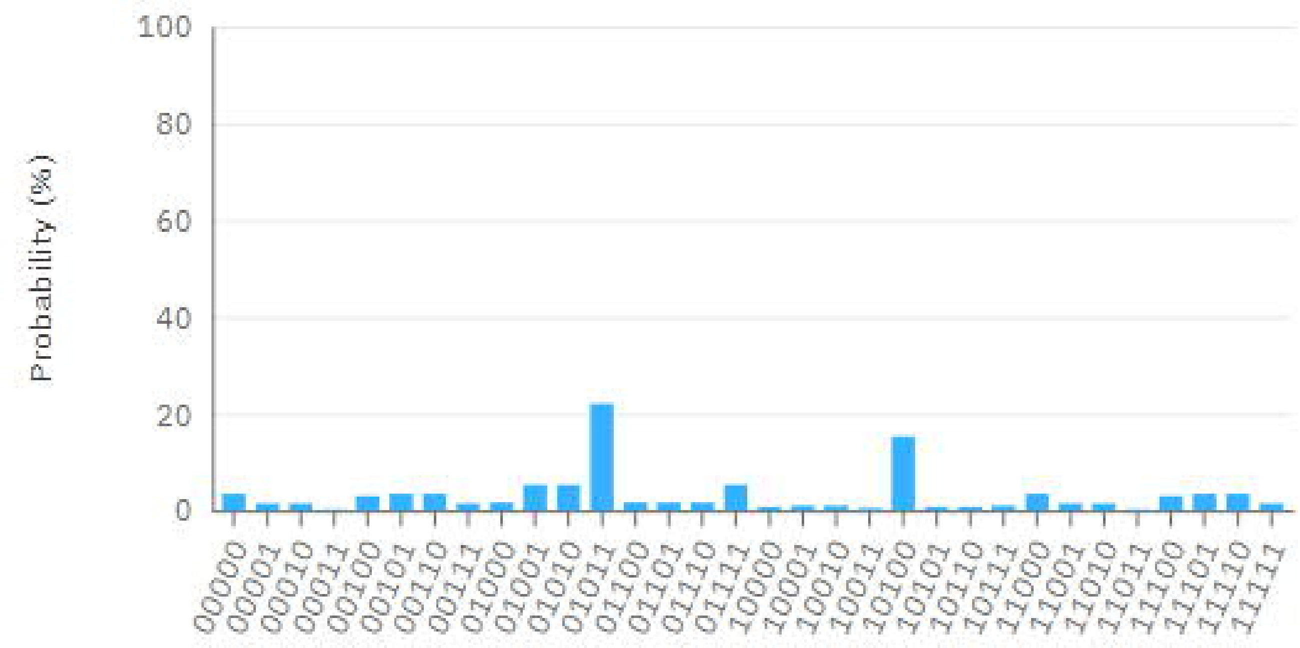



V1
V2

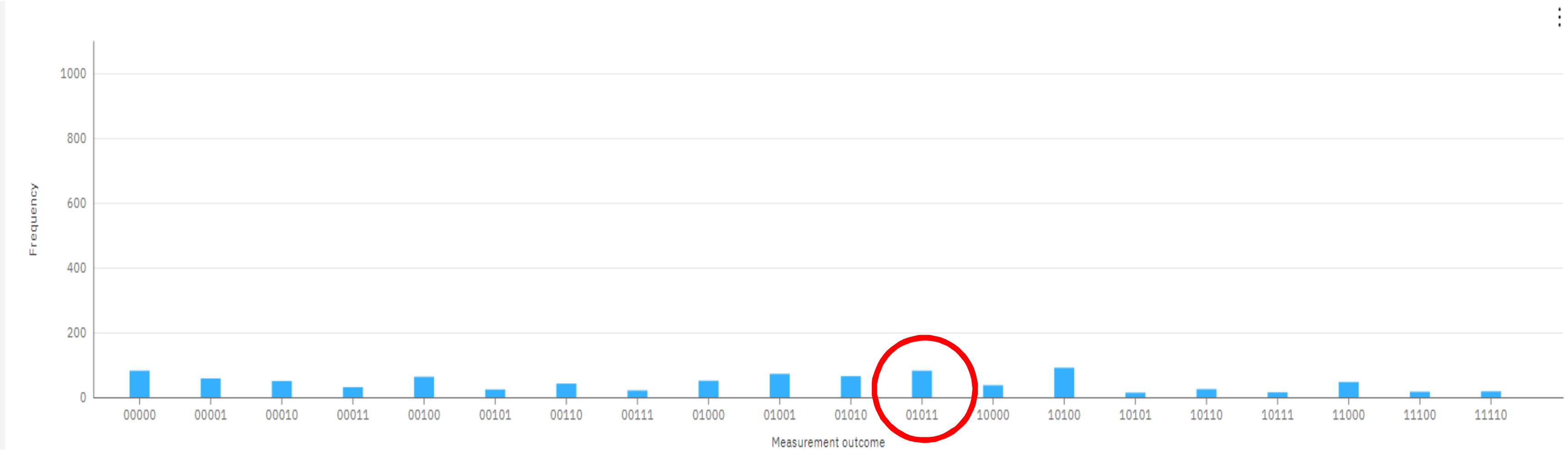

V1
V2
X
Y
V1 V2
11010


q0 H
q1 H
q2 H
q3 H
q4 H
PH
0.5 * -(tau0)
PH
-0.5 * -(tau0)
RX
(t0)
PH
0.5 * -(tau0)
PH
-0.5 * -(tau0)
RX
(t0)
PH
-0.5 * -(tau0)
PH
0.5 * -(tau0)
RX
(t0)
RX
(t0)
RX
(t0)

IBM simulator



IBM Lima

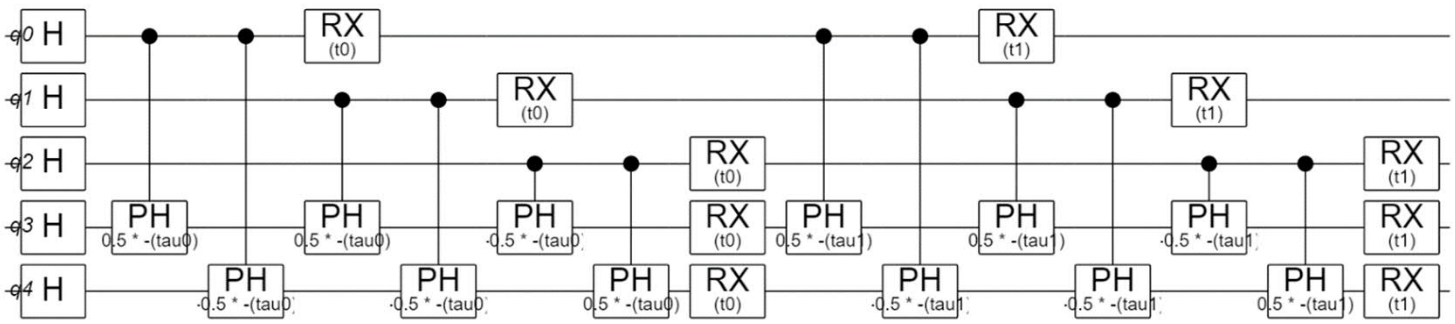


Frequency
1000
800
600
400
200
0
00000
00001
00010
00011
00100
00101
00110
00111
01000
01001
01010
01011
10000
10100
10101
10110
10111
11000
11100
11110
Measurement outcome

IBM Belem

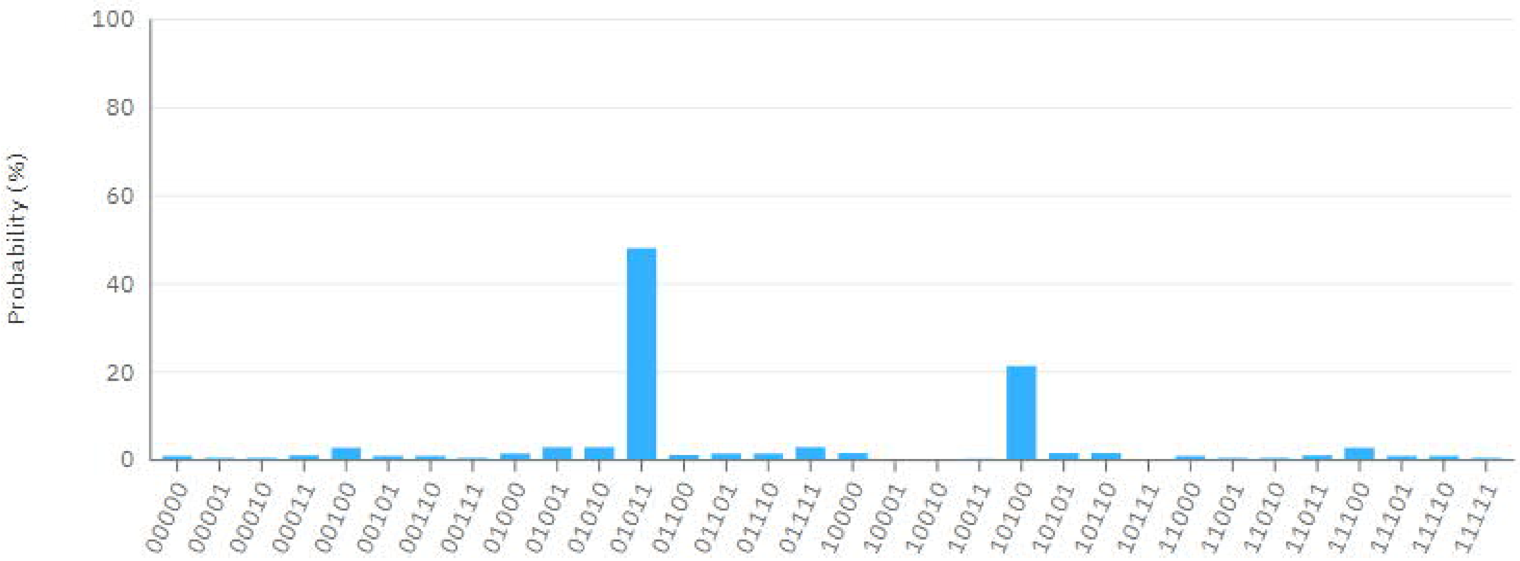
Frequency
1000
800
600
400
200
0
00000
00001
00010
00011
00100
00101
00110
01000
01001
01010
01011
01100
01110
10000
10100
10110
11000
11001
11011
11100
Measurement outcome

Two rounds, p=2

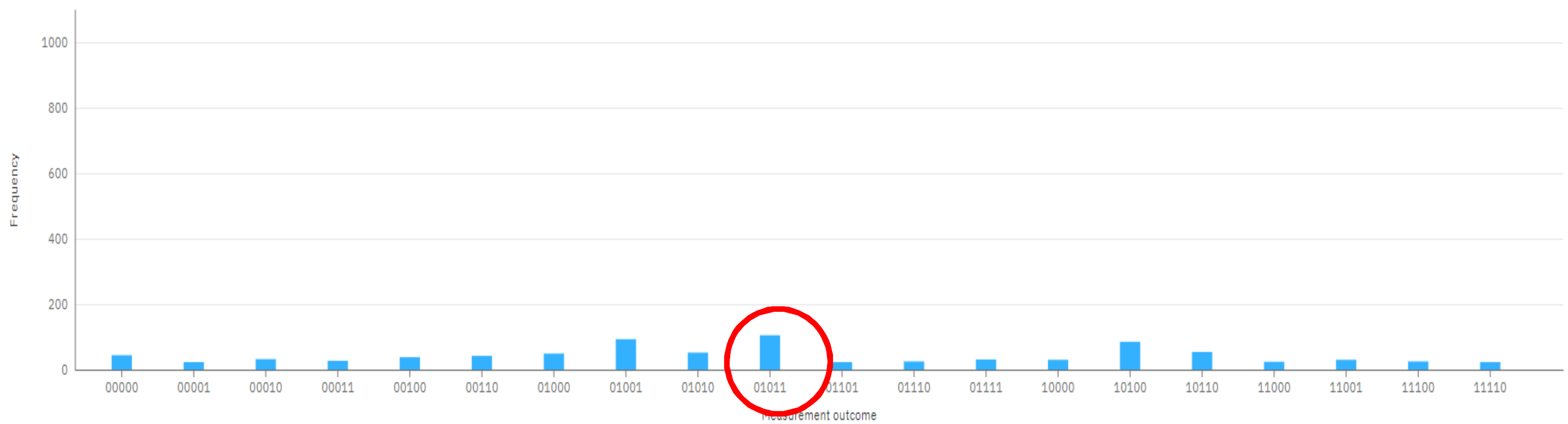


Optimising n=3, m=2, p=2.
Method = Powell
elapsed time: 0:00:03.839002
Time per function evaluation 0:00:00.009363
tau0 = 10.116889479098923, t0 = 11.471736256440398
tau1 = 9.17638190857584, t1 = 5.579403784536778
Best energy: -0.63
State |11010> with prob 0.481
State |00101> with prob 0.235
State |00111> with prob 0.027

IBM simulator



IBM Athens



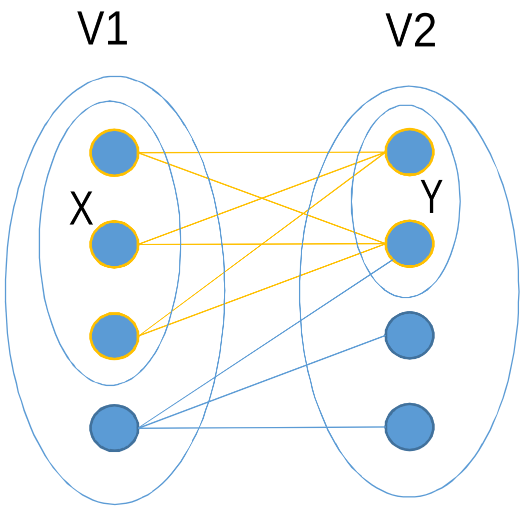
Frequency
1000
800
600
400
200
0
00000
00001
00010
00011
00100
00110
01000
01001
01010
01011
01101
01110
01111
10000
10100
10110
11000
11001
11100
11110

More rounds?

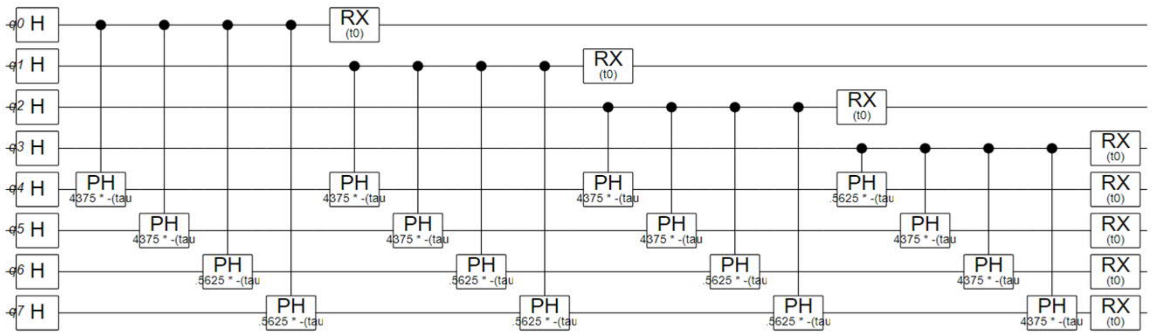
Optimising n=3, m=2, p=3.
Method = Powell
elapsed time: 0:00:05.946319
Time per function evaluation 0:00:00.010288
tau0 = 1.9512700487624643, t0 = 2.35804574941005
tau1 = 1.5414280095437825, t1 = 1.6301424630590888
tau2 = 2.7728445061690166, t2 = 1.055323840771294
Best energy: -0.557
State |11010> with highest prob 0.353

Optimising n=3, m=2, p=4.
Method = Powell
elapsed time: 0:00:11.298022
Time per function evaluation 0:00:00.011264
tau0 = -2.2575416257019723, t0 = 0.8509021588967012
tau1 = 1.3366706970924482, t1 = 1.1689045404771015
tau2 = -4.612107452590686, t2 = 0.9580506935056163
tau3 = 1.278055613216713, t3 = 1.4458889821412686
Best energy: -0.6815000000000001
State |11010> with highest prob 0.532

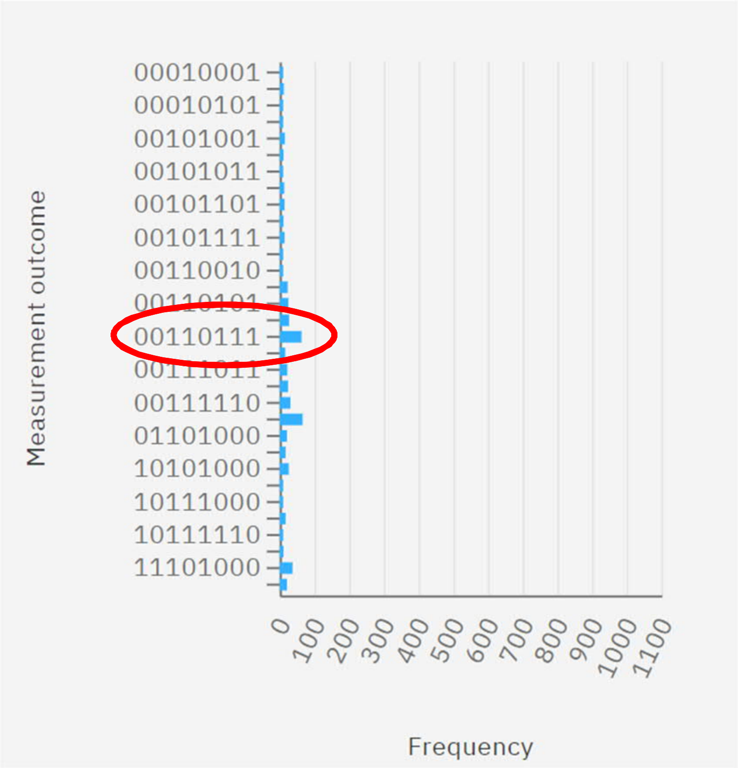
Optimising n=3, m=2, p=5.
Method = Powell
elapsed time: 0:00:21.385700
Time per function evaluation 0:00:00.012014
tau0 = -1.4699400667586102, t0 = 2.126774293106693
tau1 = 0.4385289326847269, t1 = 0.974445950989624
tau2 = 2.446442361429776, t2 = 1.402996346891495
tau3 = 2.421268443405987, t3 = 1.2375223517742582
tau4 = 2.8185891764310496, t4 = 0.74636509943765
Best energy: -0.914
State |11010> with highest prob 0.88


V1
V2
X
Y

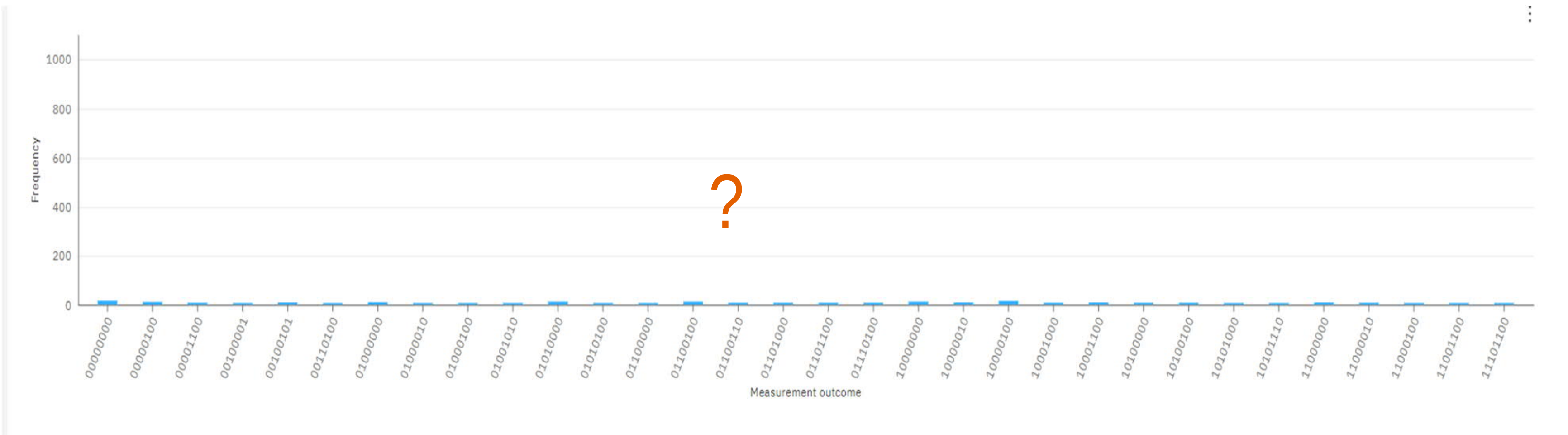
11101100


q0 H
q1 H
q2 H
q3 H
q4 H
q5 H
q6 H
q7 H
PH
4375 * -(tau
PH
5625 * -(tau
RX
(t0)

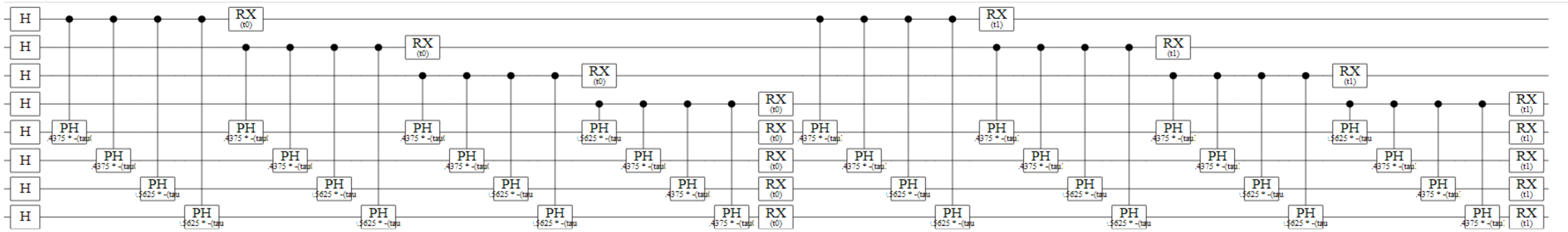
IBM simulator



IBM Melbourne (16 qubits)

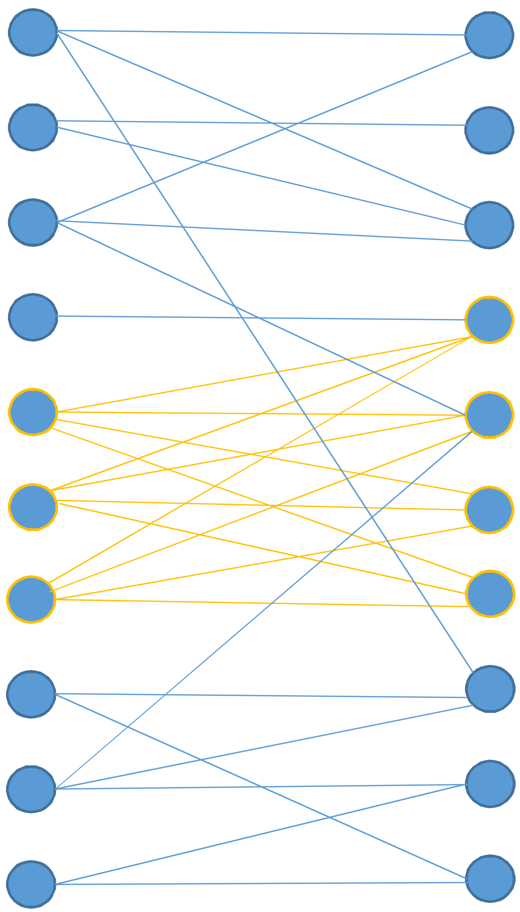


More rounds

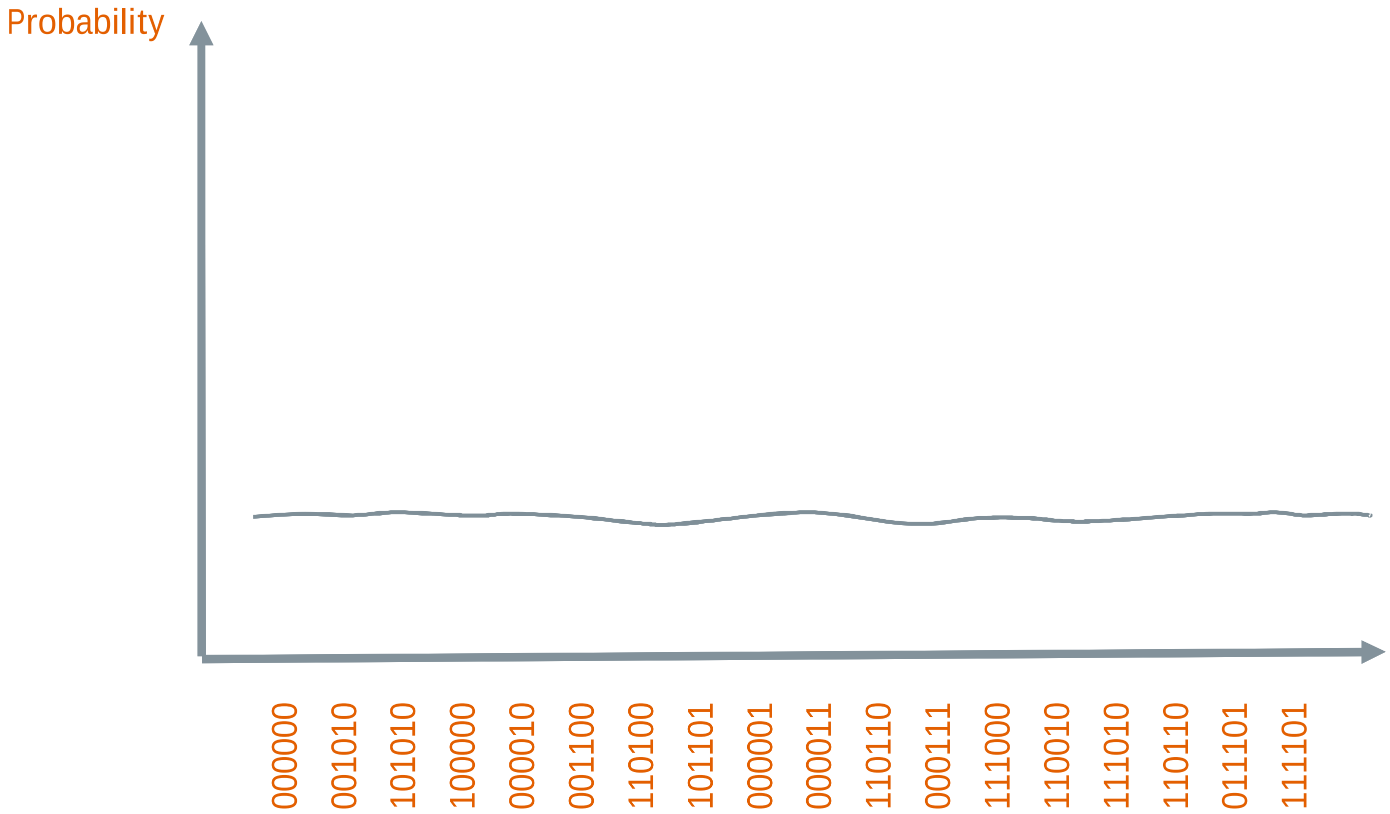


p=2
Best energy = -1.660750
State |11111100> with prob 0.258 , 66.048 times higher than equal prob 0.00390625.
State |11101100> with prob 0.137 , 35.072 times higher than equal prob 0.00390625.
State |00010111> with prob 0.046 , 11.776 times higher than equal prob 0.00390625.

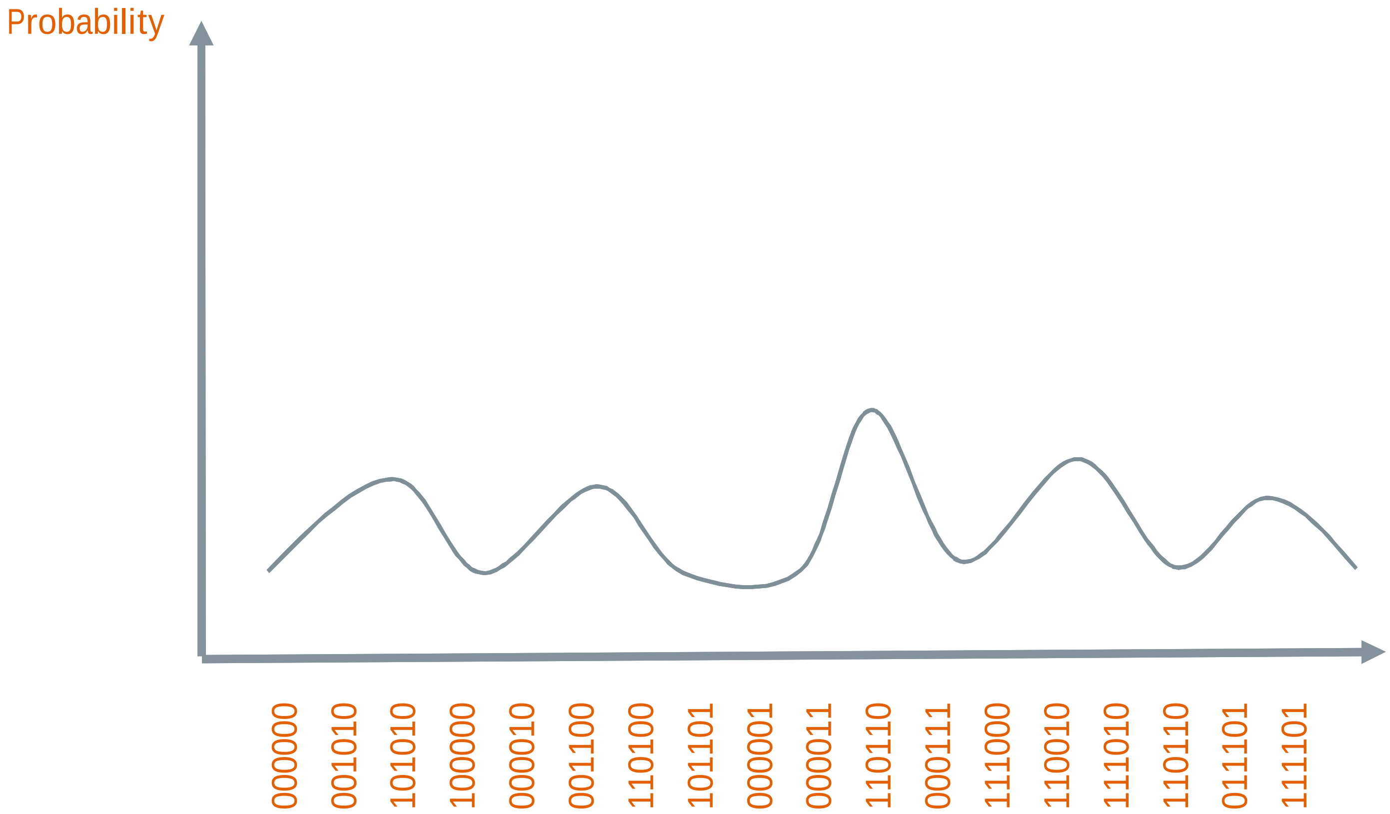
p=5
Best energy = -1.997125
State |11111100> with prob 0.406 , 103.936 times higher than equal prob 0.00390625.
State |11101100> with prob 0.188 , 48.128 times higher than equal prob 0.00390625.
State |11110100> with prob 0.032 , 8.192 times higher than equal prob 0.00390625.

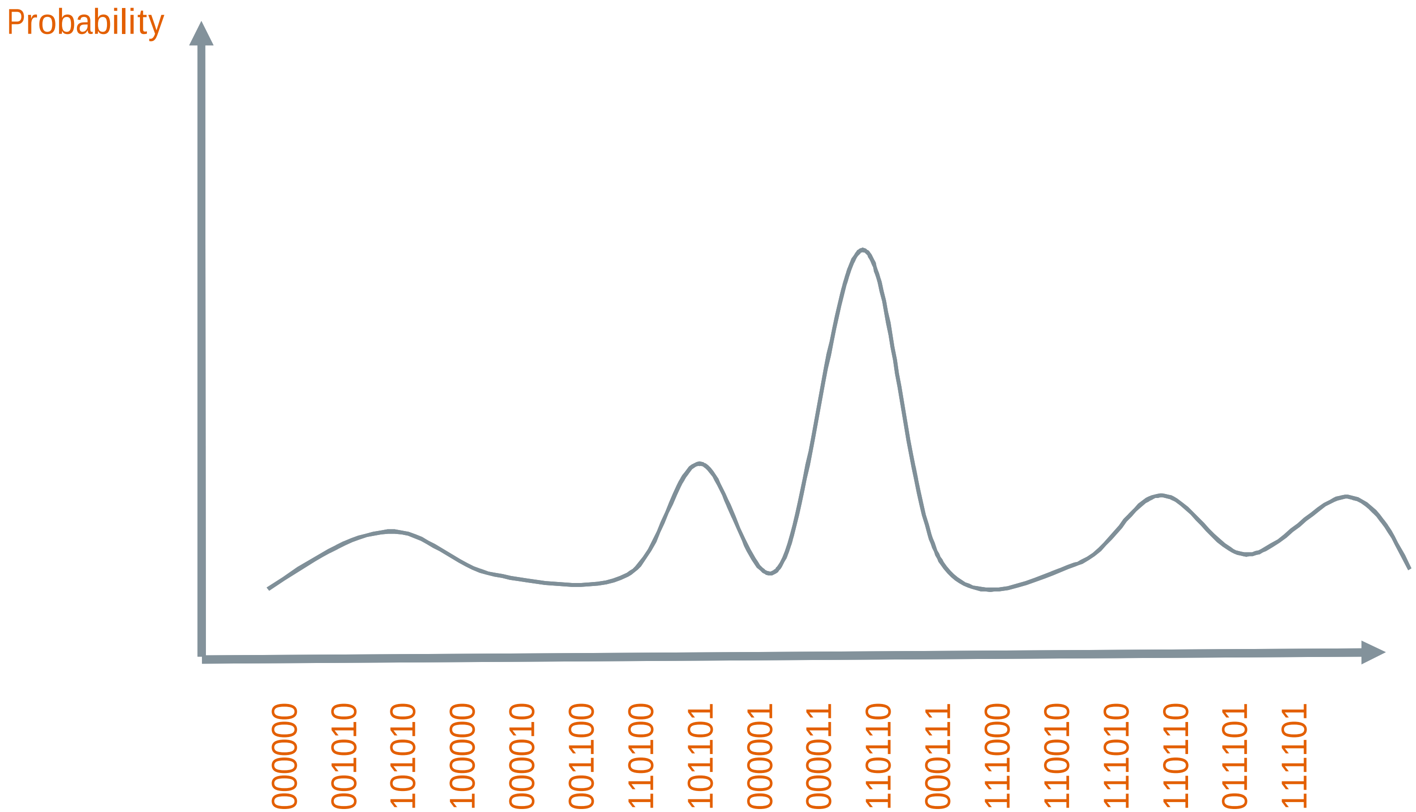


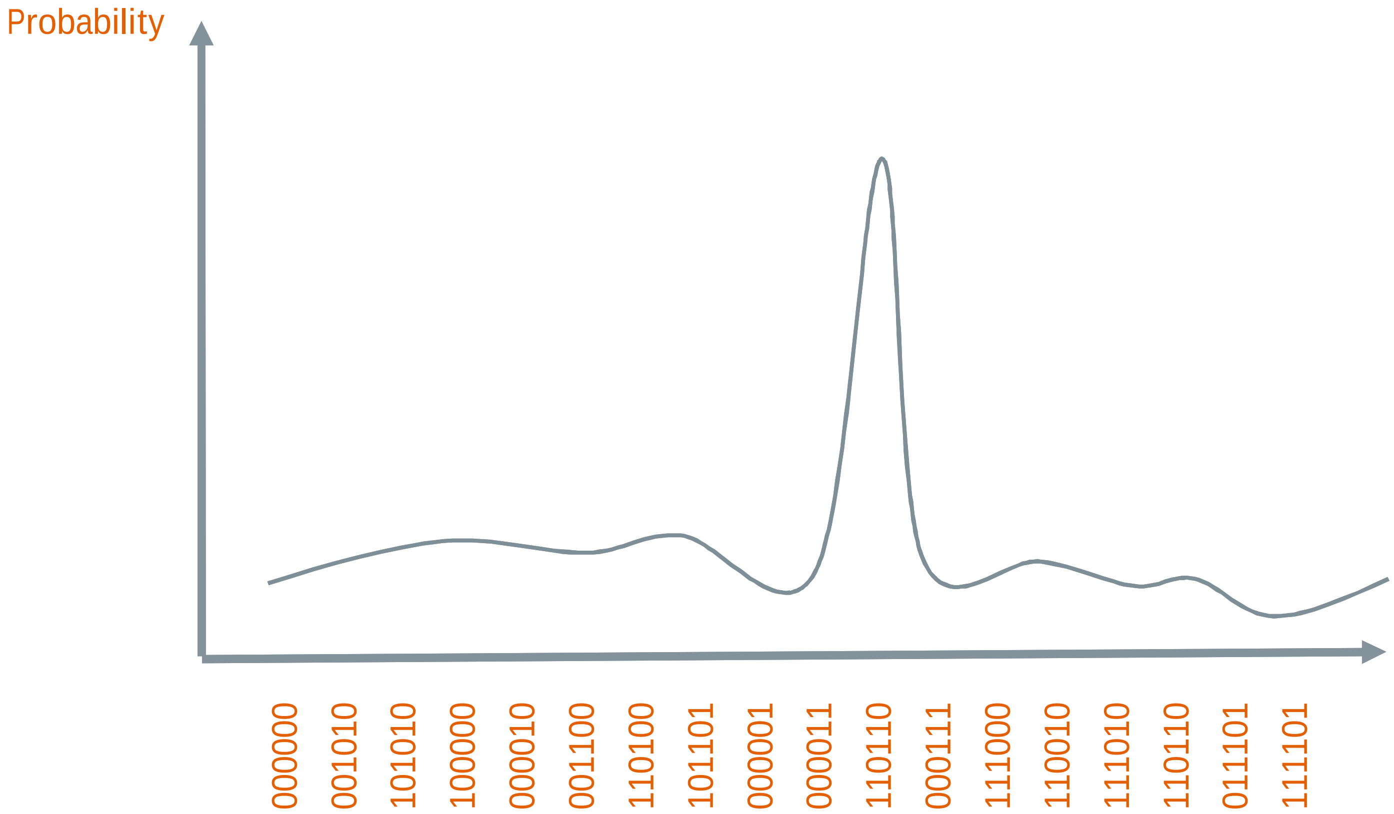
Two rounds
Best energy = -3.753360
tau0 = 1.1941295703091928, t0 = 13.60773822086078
tau1 = 3.8860783390590563, t1 = 18.842787584395566
State |00001110000001111000> of energy 8.64 with prob 0.004, 4194.304 times equal prob 9.5367431640625e-07.
State |10101110001011111000> of energy 8.6 with prob 0.004, 4194.304 times equal prob 9.5367431640625e-07.
State |10101110101011111100> of energy 8.24 with prob 0.004, 4194.304 times equal prob 9.5367431640625e-07



Probability
000000
001010
101010
100000
000010
001100
110100
101101
000001
000011
110110
000111
111000
110010
111010
110110
011101
111101

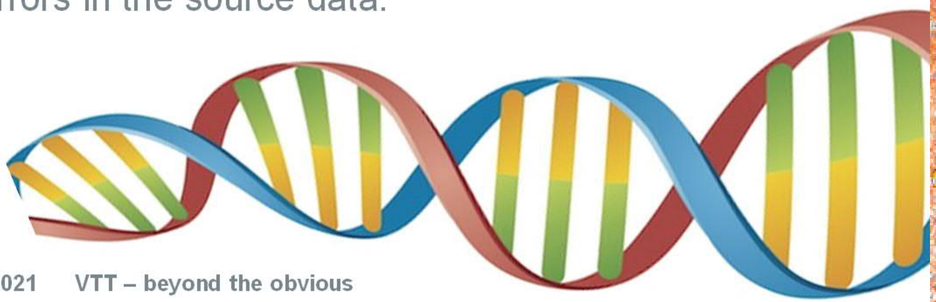

Probability
000000
001010
101010
100000
000010
001100
110100
101101
000001
000011
110110
000111
111000
110010
111010
110110
011101
111101

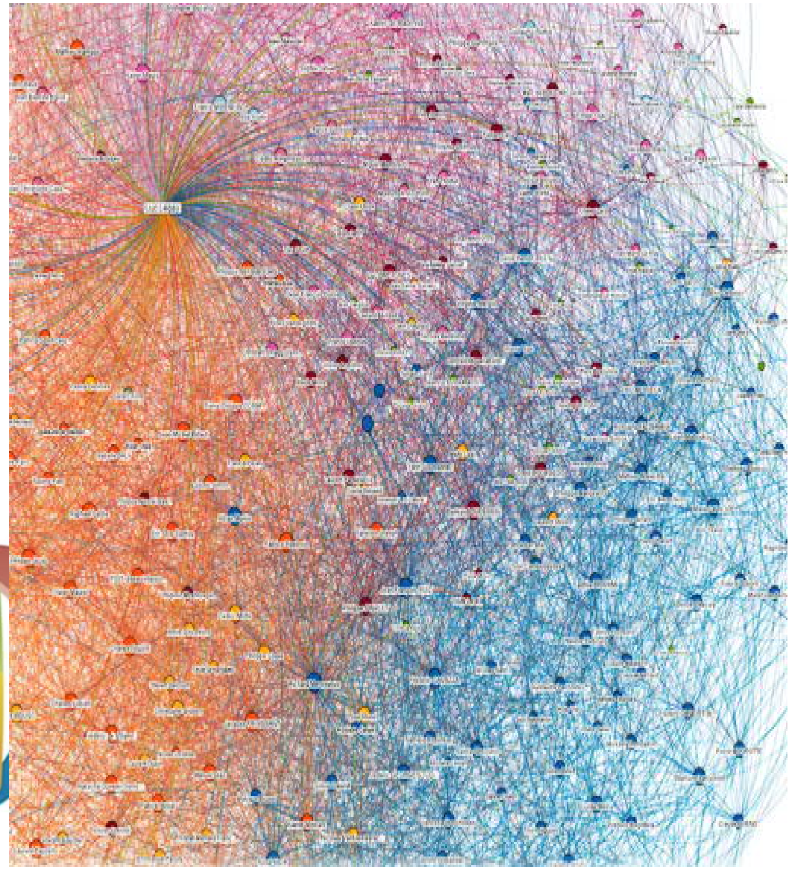

Probability
000000
001010
101010
100000
000010
001100
110100
101101
000001
000011
110110
000111
111000
110010
111010
110110
011101
111101


Probability
000000
001010
101010
100000
000010
001100
110100
101101
000001
000011
110110
000111
111000
110010
111010
110110
011101
111101

Applying community panning to other domains

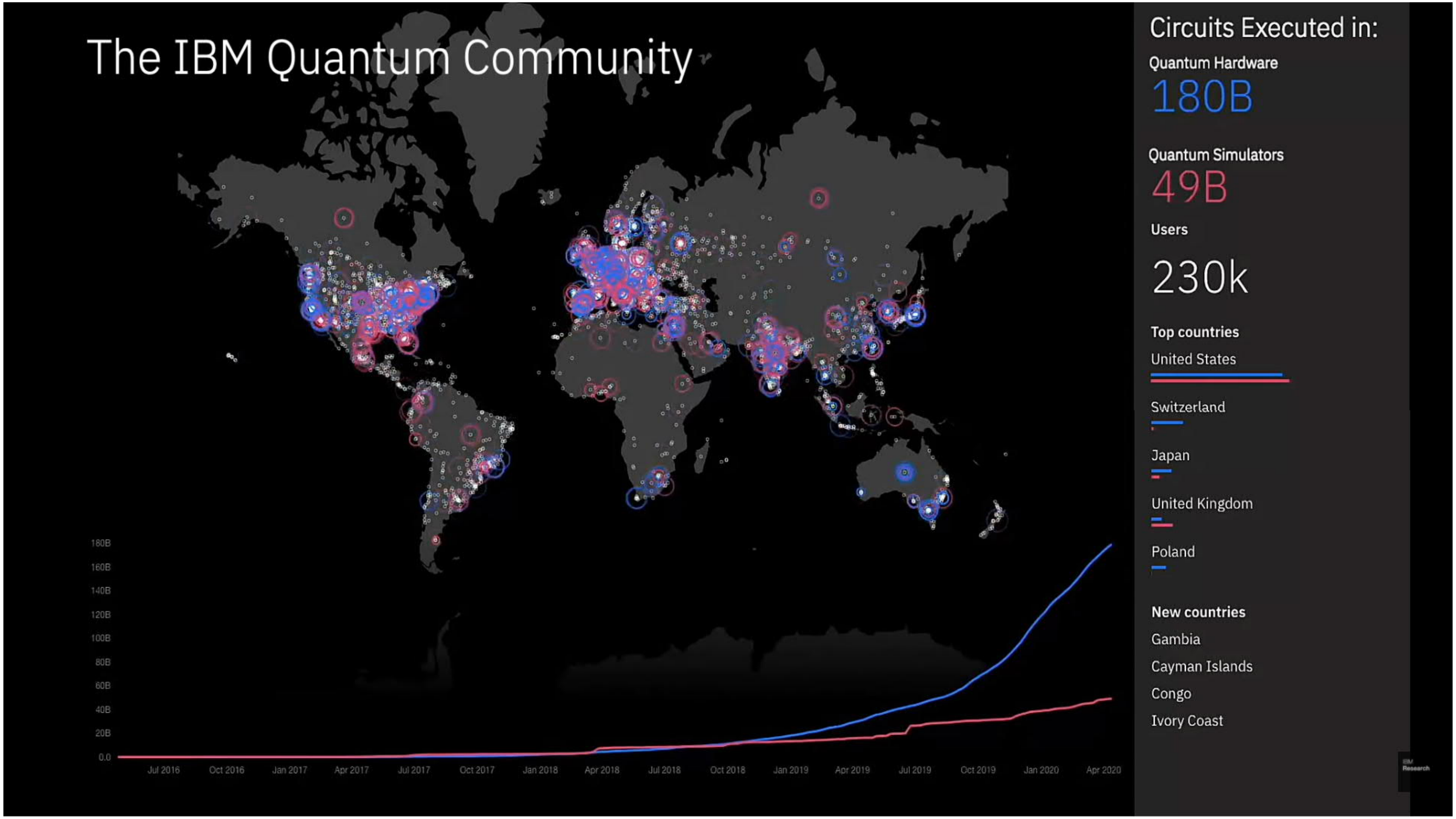
- De novo clustering of metagenomics data
- Metagenomics data of a sample can be translated to a graph for finding clusters that are expected to correspond to the species present in the sample.
- However, the clusters may or may not correspond to known microbes, thus providing a way to find novel species.
- Classical approaches exist, but they are CPU-intensive, new algorithms are needed for speed up and to tolerate the errors in the source data.



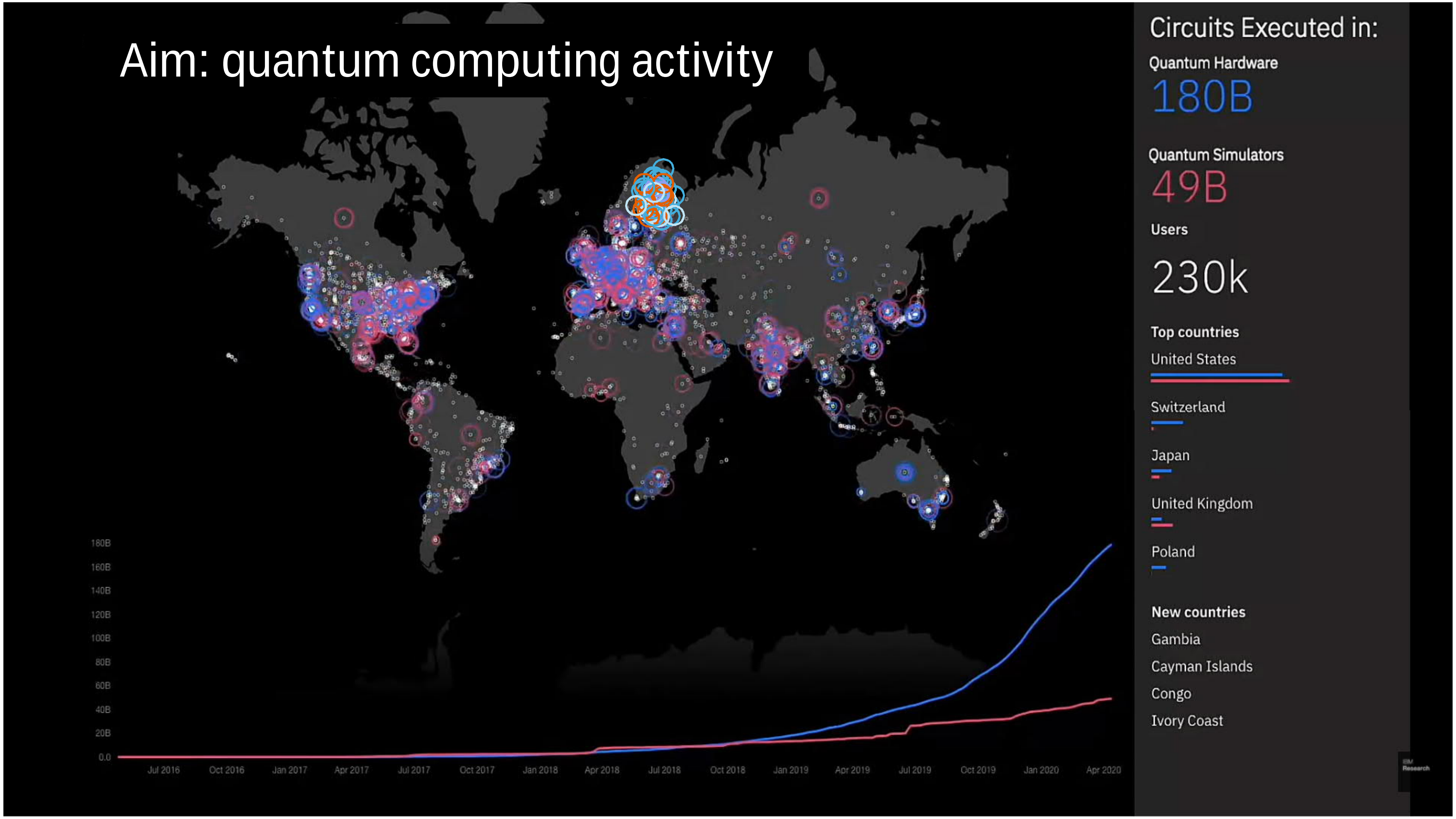


Conclusions

- In addition to learning the basics of quantum computing and existing algorithms, we need to focus on developing quantum algorithms
- Especially, for something useful, we need to work with domain experts with knowhow on state-of-the-art classical algorithms
- VTT is preparing projects in the field, please, let's join forces and change the world !


The IBM Quantum Community
Circuits Executed in:
Quantum Hardware
180B
Quantum Simulators
49B
Users
230k
Top countries
United States
Switzerland
Japan
United Kingdom
Poland
New countries
Gambia
Cayman Islands
Congo
Ivory Coast
180B
160B
140B
120B
100B
80B
60B
40B
20B
0.0
Jul 2016
Oct 2016
Jan 2017
Apr 2017
Jul 2017
Oct 2017
Jan 2018
Apr 2018
Jul 2018
Oct 2018
Jan 2019
Apr 2019
Jul 2019
Oct 2019
Jan 2020
Apr 2020

Aim: quantum computing activity



beyond

the obvious

First Name Surname
firstname.surname@vtt.fi
+358 1234 5678

@VTTFinland
@your_account

www.vtt.fi