

“Introduction to Quantum Computing - Optimizing Logistics Case Study”

Paweł Gora
Fundacja Quantum AI

3.04.2025



What is quantum computing?

(Too) Simple answer: “Computations according to the laws of quantum mechanics”

R. Feynmann “There's Plenty of Room at the Bottom” (1960):

“There is nothing that I can see in the physical laws that says the computer elements cannot be made enormously smaller than they are now. In fact, there may be certain advantages.”

Moore's law

(Gordon) Moore's law: “The number of transistors in a dense integrated circuit doubles about every two years.”

Moore's prediction proved accurate for several decades and has been used in the semiconductor industry to guide long-term planning and to set targets for research and development.

Moore's law

(Gordon) Moore's law: “The number of transistors in a dense integrated circuit doubles about every two years.”

Moore's prediction proved accurate for several decades and has been used in the semiconductor industry to guide long-term planning and to set targets for research and development.

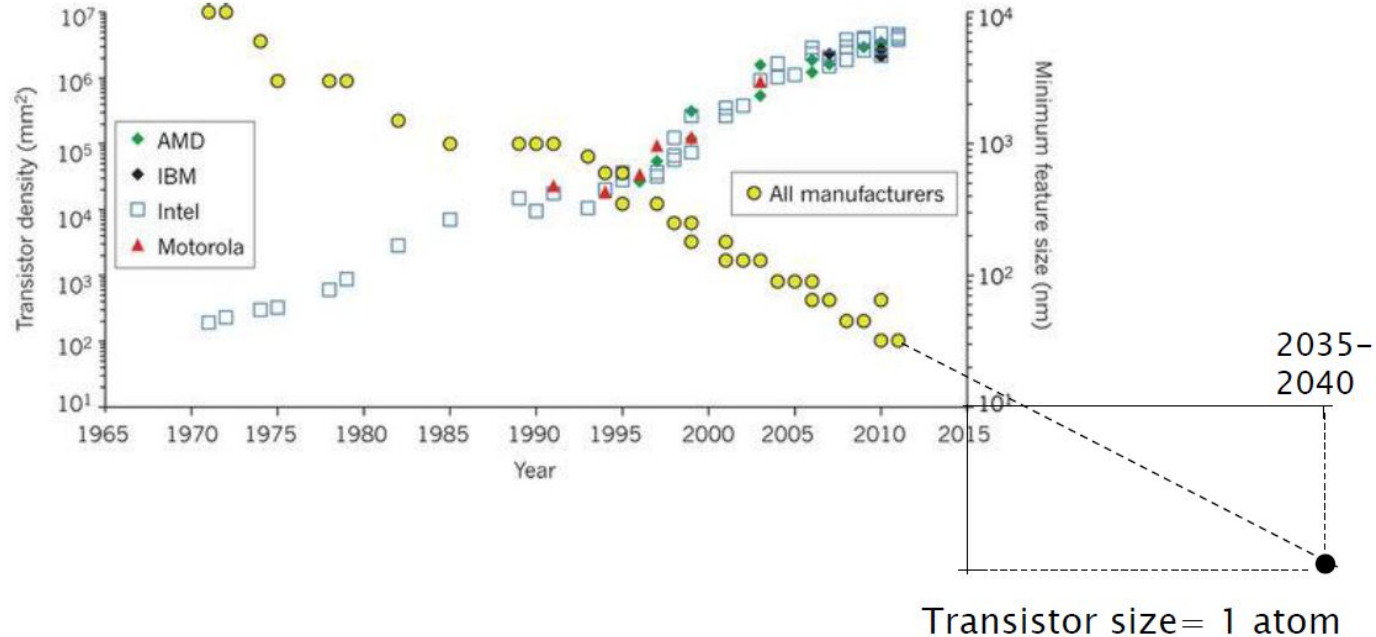
Moore 2015: “I see Moore's law dying here in the next decade or so.”

Intel:

- 2015: the pace of advancement has slowed (two years -> 2.5 year)
- 2017: the trend will continue thanks to hyperscaling

Moore's law

(Gordon) Moore's law: “The number of transistors in a dense integrated circuit doubles about every two years.”

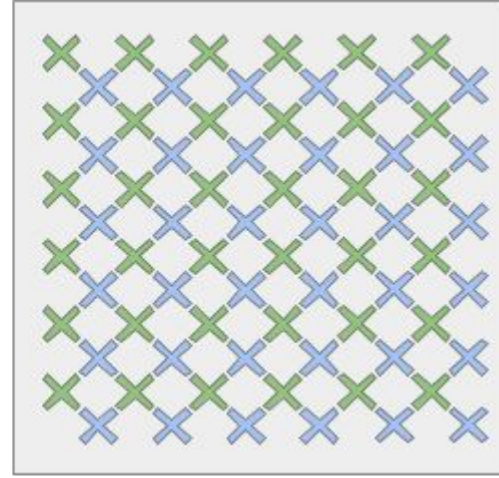
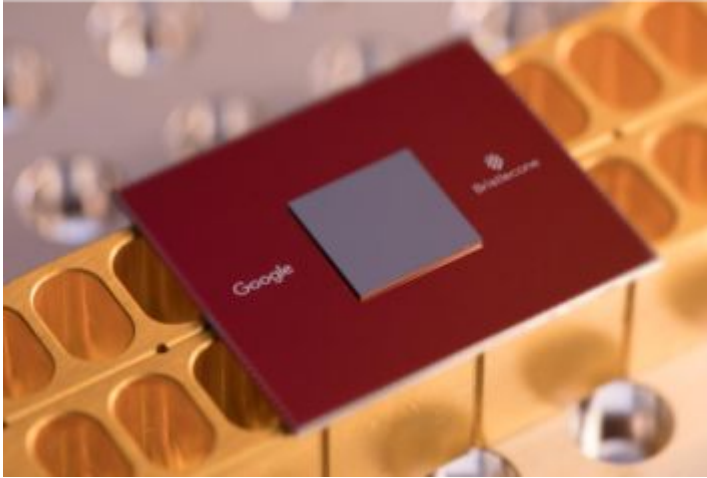


World is quantum, so is computing

R. Feynmann “Tiny Computers Obeying Quantum Mechanical Laws” (1983):

“Now, we can, in principle make a computing device in which the numbers are represented by a row of atoms with each atom in either of the two states. That’s our input. The Hamiltonian starts “Hamiltonianizing” the wave function. . . . The ones move around, the zeros move around . . Finally, along a particular bunch of atoms, ones and zeros . . . occur that represent the answer. Nothing could be made smaller . . . Nothing could be more elegant. No losses, no uncertainties, no averaging. But can we do it?”

What is a quantum computer?

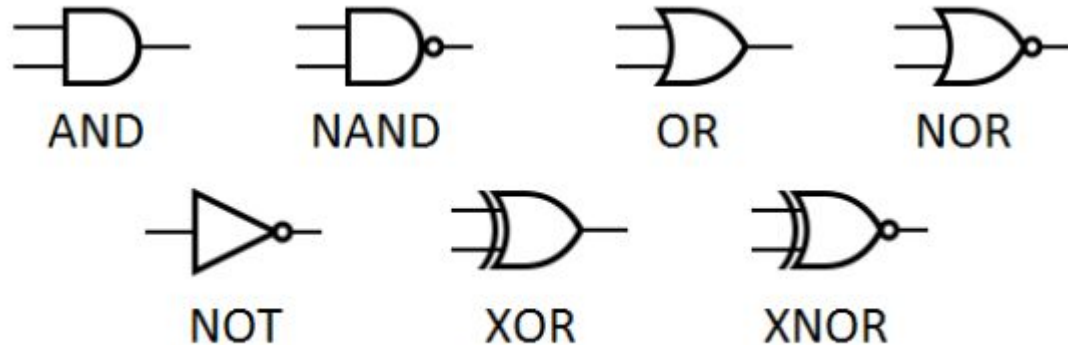


Bristlecone is one the Google's quantum processors (left). On the right is a cartoon of the device: each "X" represents a qubit, with nearest neighbor connectivity. Source:

<https://ai.googleblog.com/2018/03/a-preview-of-bristlecone-googles-new.html>

What is a quantum computer?

Classical computers	Quantum computers
Bits and classical gates	Qubits and quantum gates



Source: <https://blog.digilentinc.com/logic-gates>

What is a quantum computer?

Classical computers	Quantum computers
Bits and classical gates	Qubits and quantum gates
Bit: state has a value 0 or 1 (mathematically: element from the set $\{0,1\}$)	Qubit: state may be in a superposition of states 0 and 1 (mathematically: a unitary vector of the 2-dimensional Hilbert space over complex numbers)

What is a quantum computer?

Classical computers	Quantum computers
Bits and classical gates	Qubits and quantum gates
Bit: state has a value 0 or 1 (mathematically: element from the set $\{0,1\}$)	Qubit: state may be in a superposition of states 0 and 1 (mathematically: a unitary vector of the 2-dimensional Hilbert space over complex numbers)
State is determined (we can know exactly whether the state is 0 or 1) and can be changed from 0 to 1 and vice versa	To get values 0 or 1, we make a measurement that breaks the superposition - the probability of getting 0 (or 1) depends on the state of the qubit (unitary vector...), quantum gates change qubits (and the probabilities).

What is a qubit?

$$|\psi\rangle = a|0\rangle + b|1\rangle$$



Quantum superposition

$|0\rangle$ - Bra-ket notation (Dirac notation)

$|0\rangle$ and $|1\rangle$ are basis states (vectors), an orthonormal base of a quantum system ($|0\rangle = (1, 0)$, $|1\rangle = (0, 1)$)

a, b are complex numbers, squares of their amplitudes ($|a|^2$, $|b|^2$) are probabilities of getting 0 or 1 in a measurement ($|a|^2 + |b|^2 = 1$).

What is a qubit?

$$|\psi\rangle = a|0\rangle + b|1\rangle$$



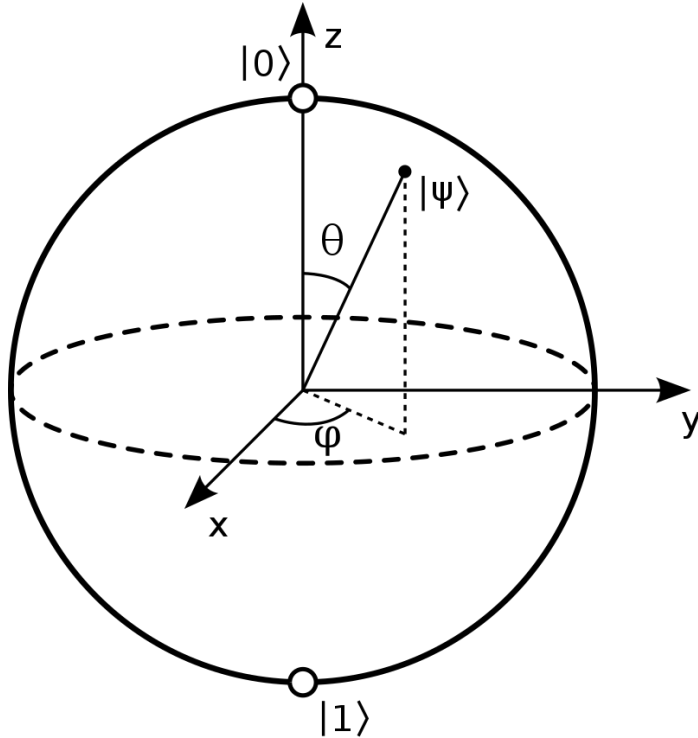
Quantum superposition

$|0\rangle$ - Bra-ket notation (Dirac notation)

$|0\rangle$ and $|1\rangle$ are basis states (vectors), an orthonormal base of a quantum system ($|0\rangle = (1, 0)$, $|1\rangle = (0, 1)$)

a, b are complex numbers, squares of their amplitudes ($|a|^2$, $|b|^2$) are probabilities of getting 0 or 1 in a measurement ($|a|^2 + |b|^2 = 1$), multiplying them by any unitary vector doesn't change the quantum state, a, b can be transformed to spherical coordinates ...

Bloch sphere



In classical computers:

bit's state can be 0 or 1

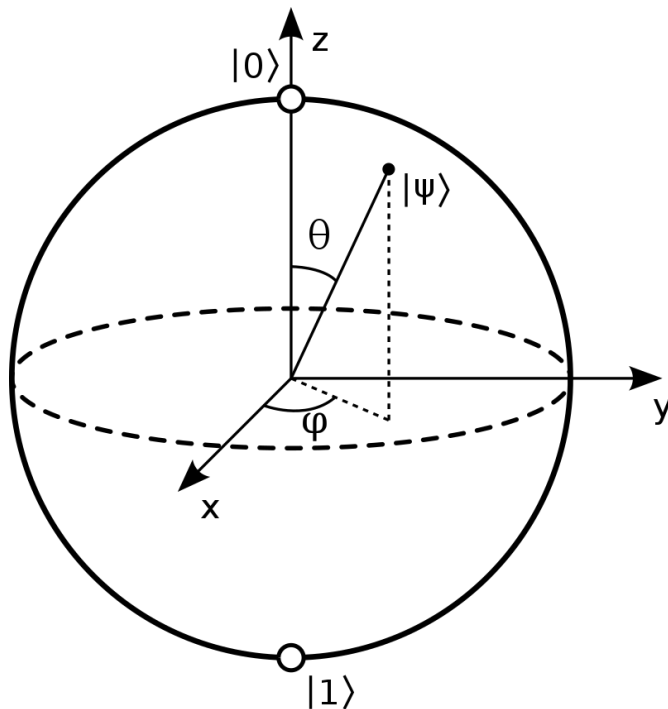
In quantum computers:

qubit's state can be any point on the Bloch sphere

Source: https://en.wikipedia.org/wiki/Bloch_sphere

$$|\psi\rangle = \cos(\theta/2)|0\rangle + e^{i\phi} \sin(\theta/2)|1\rangle = \cos(\theta/2)|0\rangle + (\cos\phi + i\sin\phi) \sin(\theta/2)|1\rangle, \text{ where } 0 \leq \theta < \pi \text{ and } 0 \leq \phi < 2\pi.$$

Quantum Computing - a journey on a Bloch sphere



Source: https://en.wikipedia.org/wiki/Bloch_sphere

$$|\psi\rangle = \cos(\theta/2)|0\rangle + e^{i\phi} \sin(\theta/2)|1\rangle = \cos(\theta/2)|0\rangle + (\cos \phi + i \sin \phi) \sin(\theta/2)|1\rangle, \text{ where } 0 \leq \theta < \pi \text{ and } 0 \leq \phi < 2\pi.$$

Quantum gates

Notable examples:

- H - Hadamard's gate:
 - ◆ $|0\rangle \rightarrow (|0\rangle + |1\rangle) / \sqrt{2}$
 - ◆ $|1\rangle \rightarrow (|0\rangle - |1\rangle) / \sqrt{2}$
- X - Pauli-X gate:
 - ◆ $|0\rangle \rightarrow |1\rangle$
 - ◆ $|1\rangle \rightarrow |0\rangle$
- Y - Pauli-Y gate:
 - ◆ $|0\rangle \rightarrow i|1\rangle$
 - ◆ $|1\rangle \rightarrow -i|0\rangle$
- Z - Pauli-Z gate:
 - ◆ $|0\rangle \rightarrow |0\rangle$
 - ◆ $|1\rangle \rightarrow -|1\rangle$
- $R_y(\phi)$ - (parametrized) rotation gate:
 - ◆ $|0\rangle \rightarrow \cos(\phi/2)|0\rangle - \sin(\phi/2)|1\rangle$
 - ◆ $|1\rangle \rightarrow \sin(\phi/2)|0\rangle + \cos(\phi/2)|1\rangle$
- CNOT: 2 qubits, one qubit is a controller for NOT (X) on second qubit

Any quantum circuit can be simulated to an arbitrary degree of accuracy using a combination of CNOT gates and single qubit rotations.

Quantum gates

Notable examples:

→ H - Hadamard's gate:

- ◆ $|0\rangle \rightarrow (|0\rangle + |1\rangle) / \sqrt{2}$
- ◆ $|1\rangle \rightarrow (|0\rangle - |1\rangle) / \sqrt{2}$



It represents a rotation of π about the axis $\mathbf{X+Z}$ (alternatively: $\pi/2$ around \mathbf{Y} -axis and then π around \mathbf{Z} -axis).

Hadamard matrix:
$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

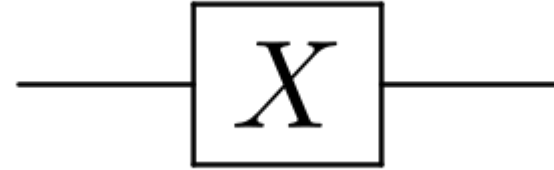
Quantum gates

Notable examples:

→ X - Pauli-X gate:

◆ $|0\rangle \rightarrow |1\rangle$

◆ $|1\rangle \rightarrow |0\rangle$



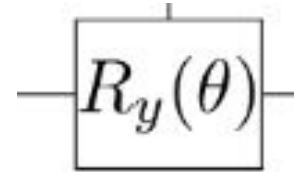
It is a quantum equivalent of the NOT gate. It represents a rotation of π about the axis \mathbf{X} .

Pauli X matrix:
$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Quantum gates

Notable examples:

- $R_y(\phi)$ - (parametrized) rotation gate:
- ◆ $|0\rangle \rightarrow \cos(\phi/2)|0\rangle - \sin(\phi/2)|1\rangle$
 - ◆ $|1\rangle \rightarrow \sin(\phi/2)|0\rangle + \cos(\phi/2)|1\rangle$



It represents a rotation through angle ϕ (radians) around the y-axis.

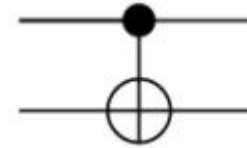
$R_y(\phi)$ matrix:

$$R_y(\theta) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix}$$

Quantum gates

Notable examples:

→ CNOT gate



CNOT acts on 2 qubits, and performs the NOT operation on the second qubit only when the first qubit is $|1\rangle$, and otherwise leaves it unchanged.

CNOT matrix:
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Here:

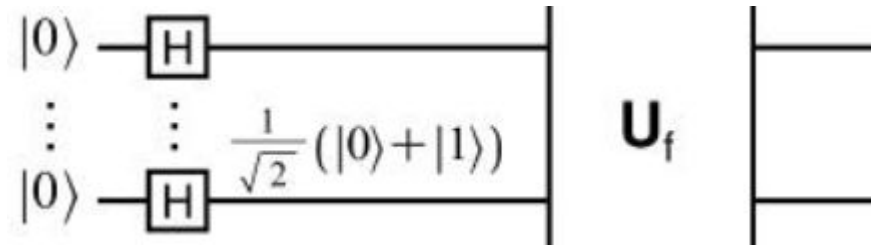
$|00\rangle \rightarrow [1, 0, 0, 0]$

$|01\rangle \rightarrow [0, 1, 0, 0]$

$|10\rangle \rightarrow [0, 0, 1, 0]$

$|11\rangle \rightarrow [0, 0, 0, 1]$

Quantum parallelism



N qubits prepared as a superposition of 2^N numbers.

$$|\psi\rangle = |00 \dots 0\rangle + |00 \dots 1\rangle + \dots + |11 \dots 1\rangle$$

In a single run we process all states present in the superposition.

$$|\psi'\rangle = U|00 \dots 0\rangle + U|00 \dots 1\rangle + \dots + U|11 \dots 1\rangle$$

Why it is important?

Classical computers	Quantum computers
On n bits we can process only one n -bit number at the same time	Using n qubits we can process all 2^n n -bit numbers at the same time

Why it is important?

Classical computers	Quantum computers
On n bits we can process only one n -bit number at the same time	Using n qubits we can process all 2^n n -bit numbers at the same time
Results are deterministic, randomness is a pseudo-randomness	We may have real randomness and easily sample from probability distributions which are difficult for classical computers

Applications

- Breaking (some) classical cryptography algorithms
- Solving combinatorial optimization problems (AI, finance, transport / logistics)
- Sampling from difficult distributions (e.g., Boltzmann distribution -> applications in ML), quantum random number generation
- Simulating quantum processes (discovering new materials, drugs etc)
- Searching in databases
- Generating art (e.g., music)

There are also applications of other quantum technologies:

- Quantum communication / quantum Internet / quantum key distribution
- Quantum metrology

Adiabatic quantum computers

An **adiabatic process** - a process that does not involve the transfer of heat or matter into or out of a thermodynamic system. In an adiabatic process, energy is transferred to the surroundings only as work.

(Source: https://en.wikipedia.org/wiki/Adiabatic_process)

Adiabatic quantum computers

An **adiabatic process** - a process that does not involve the transfer of heat or matter into or out of a thermodynamic system. In an adiabatic process, energy is transferred to the surroundings only as work.

(Source: https://en.wikipedia.org/wiki/Adiabatic_process)

Adiabatic quantum computer:

“First, a (potentially complicated) Hamiltonian is found whose ground state describes the solution to the problem of interest. Next, a system with a simple Hamiltonian is prepared and initialized to the ground state. Finally, the **simple Hamiltonian is adiabatically evolved** to the desired complicated Hamiltonian. By the adiabatic theorem, the system remains in the ground state, so at the end the state of the system describes the solution to the problem.” (Source: https://en.wikipedia.org/wiki/Quantum_annealing)

Adiabatic quantum computing has been shown to be **polynomially equivalent to conventional quantum computing in the circuit model**. (“Adiabatic Quantum Computation is Equivalent to Standard Quantum Computation”, D. Aharonov et al, <https://arxiv.org/pdf/quant-ph/0405098.pdf>)

Ising Model

The Ising model of ferromagnetism traditionally used in statistical mechanics. Variables are “spin up” (\uparrow) and “spin down” (\downarrow), states that correspond to $+1$ and -1 values (atomic “spins” or magnetic dipole moments). Relationships between the spins, represented by couplings, are correlations or anti-correlations. The energy (Hamiltonian) expressed as an Ising model is as follows:

$$E_{ising}(\mathbf{s}) = \sum_{i=1}^N h_i s_i + \sum_{i=1}^N \sum_{j=i+1}^N J_{i,j} s_i s_j$$

where the linear coefficients corresponding to qubit biases (caused by external magnetic fields) are h_i , and the quadratic coefficients corresponding to coupling strengths are $J_{i,j}$.

Source: https://docs.dwavesys.com/docs/latest/c_gs_2.html

Finding the minimum value of a nonplanar Ising formulation is NP-hard problem for classical computers.

Quadratic Unconstrained Binary Optimization

Quadratic Unconstrained Binary Optimization (QUBO) problems are traditionally used in computer science. Variables are TRUE and FALSE, states that correspond to 1 and 0 values. A QUBO problem is defined using an upper-diagonal matrix Q , which is an $N \times N$ upper-triangular matrix of real weights, and x , a vector of binary variables, as minimizing the function:

$$f(x) = \sum_i Q_{i,i} x_i + \sum_{i < j} Q_{i,j} x_i x_j$$

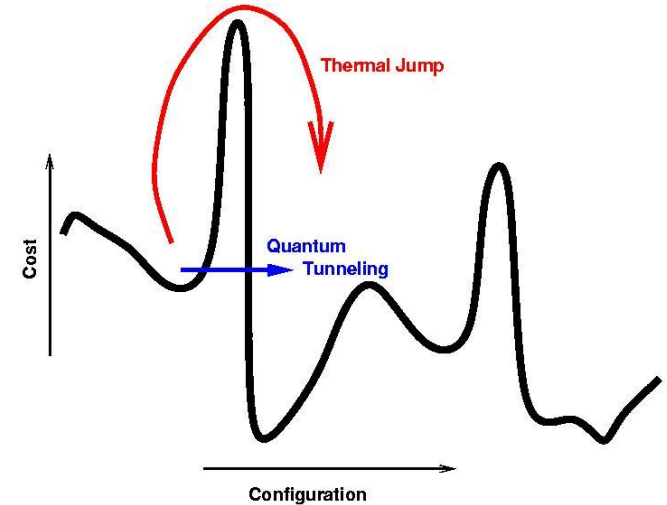
where the diagonal terms $Q_{i,i}$ are the linear coefficients and the nonzero off-diagonal terms are the quadratic coefficients $Q_{i,j}$. This can be expressed more concisely as

$$\min_{x \in \{0,1\}^n} x^T Q x$$

Quantum annealing

→ Quantum tunneling

By controlling the strength of a magnetic field (slow enough) we change the landscape: we start from the ground state of a quantum system that is easy to prepare (e.g., superposition of all possible states) and thanks to quantum tunneling we end up in a (global) minimum of a quantum state corresponding to our problem.



Source: https://en.wikipedia.org/wiki/Quantum_annealing

Nice explanation from D-Wave: https://www.youtube.com/watch?v=UV_RlCAc5Zs

Solving combinatorial optimization problems

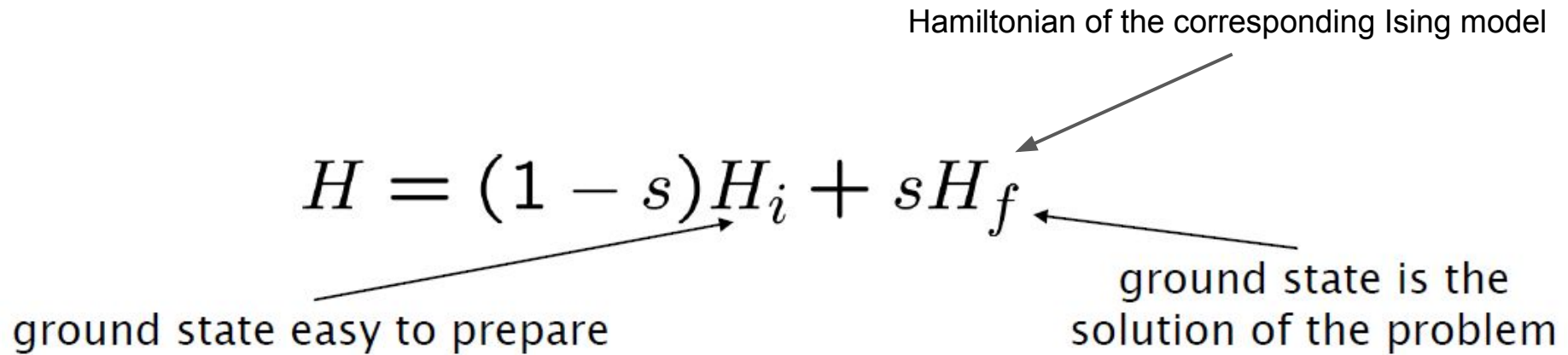
→ Quantum annealing

$$H = (1 - s)H_i + sH_f$$

Hamiltonian of the corresponding Ising model

ground state easy to prepare

ground state is the solution of the problem

The diagram shows the equation $H = (1 - s)H_i + sH_f$ centered on the page. Three arrows originate from text labels and point to specific parts of the equation: one arrow points from 'Hamiltonian of the corresponding Ising model' to H_f , another points from 'ground state easy to prepare' to H_i , and a third points from 'ground state is the solution of the problem' to H_f .

Adiabatic theorem:

if we change s **slow enough** the final state will be the solution of our problem

In practice: s can be changed by modifying a strength of the magnetic field.

Applications in transportation

Volkswagen Uses Quantum Computing to Fight Beijing Traffic

Volkswagen teamed with D-Wave Systems to run a traffic-flow algorithm on a quantum computer, with encouraging results.

BY STEPHEN EDELSTEIN MARCH 30, 2017

Volkswagen and Google to bring quantum computing benefits to cars

Posted Nov 8, 2017 by [Darrell Etherington \(@etherington\)](#)

Optimizing routes of fleets of taxis in Beijing

“Traffic Flow Optimization Using a Quantum Annealer”

Florian Neukart, Gabriele Compostella, Christian Seidel, David von Dollen, Sheir Yarkoni, Bob Parney
(Volkswagen + D-Wave)

<https://www.frontiersin.org/articles/10.3389/fict.2017.00029/full>

Quantum annealing - suitable for solving complex combinatorial optimization problems

- We build a road network and have routes from real GPS data (T-Drive, taxis in Beijing))
- For each car we add 2 possible routes between source and destination (cars may share road segments)
- Travel time is proportional to the function (square) of a number of cars on a route (simplification)
- We want to minimize the total travel time

Optimizing routes of fleets of taxis in Beijing

- q_{ij} - 0 or 1 (car i takes route j)

$$\begin{aligned} 0 &= \left(\sum_{j \in \{1,2,3\}} q_{ij} - 1 \right)^2 \\ &= -q_{i1} - q_{i2} - q_{i3} + 2q_{i1}q_{i2} + 2q_{i2}q_{i3} + 2q_{i1}q_{i3} + 1 \end{aligned}$$

- B_s - set of q_{ij} associated with routes that share street segment s

$$\text{cost}(s) = \left(\sum_{q_{ij} \in B_s} q_{ij} \right)^2$$

$$\text{Obj} = \sum_{s \in S} \text{cost}(s) + \lambda \sum_i \left(\sum_j q_{ij} - 1 \right)^2$$

Optimizing routes of fleets of taxis in Beijing

$$\text{Obj} = \sum_{s \in S} \text{cost}(s) + \lambda \sum_i \left(\sum_j q_{ij} - 1 \right)^2$$

$$\text{Obj}(x, Q) = x^T \cdot Q \cdot x$$

QUBO - Quadratic unconstrained binary optimization

Given the matrix Q , finding binary variable assignments to minimize the objective function is equivalent to minimizing an Ising model (model of ferromagnetism), which is NP-hard.

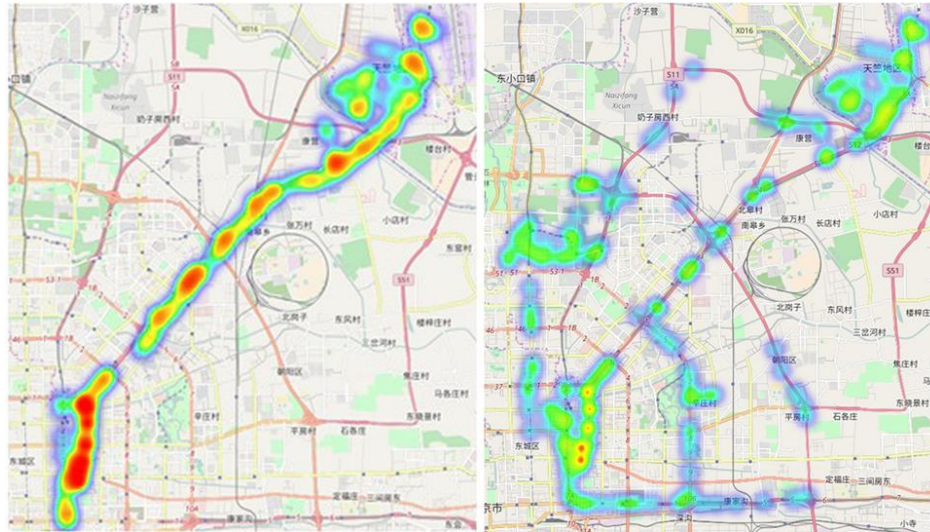
It can be solved using quantum annealing on D-Wave (in this case: D-Wave 2X QPU)

Optimizing routes of fleets of taxis in Beijing

418 cars, 1254 logical variables (3^{418} possible solutions)

Result: relatively small number of streets that are heavily occupied

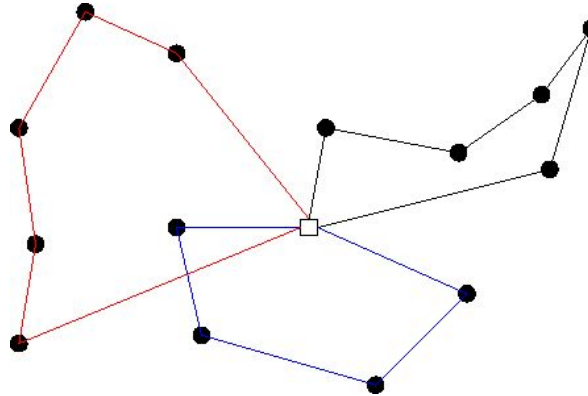
Time of computations: 22 seconds



Source: <https://www.frontiersin.org/articles/10.3389/fict.2017.00029/full>

Improving logistics

- Solving Travelling Salesman Problem and similar NP-hard problems (Vehicle Routing Problem, Pickup and Delivery problem, vanpooling)
- There are already logistic companies interested in such solutions!



- Some of my articles (the project is now supported by the Open Quantum Institute):
 - ◆ Borowski M., Gora P., Karnas K., Błajda M., Król K., Matyjasek A., Burczyk D., Szewczyk M., and Kutwin M., **"New Hybrid Quantum Annealing Algorithms for Solving Vehicle Routing Problem"**, Computational Science - ICCS 2020, 2020, pp. 546-561.
 - ◆ Nałęcz-Charkiewicz K., Das A., Chatterjee T., Keene J., Gora P., Kuhn C.C.N., **"Graph Coarsening Approach to the Vehicle Routing Problem: An Approximation Strategy"** in IEEE Access, vol. 13, pp. 22459-22472, 2025, doi: 10.1109/ACCESS.2025.3534677

Access to D-Wave (quantum annealing)

Wnioskowanie o grant pilotażowy

Wnioski na grant pilotażowy na zasoby kwantowe można składać za pośrednictwem Portalu PLGrid, wykorzystując przygotowany formularz wniosku, dostępny w [systemie grantowym](#).



Informacje o grantzie pilotażowym

- Grant pilotażowy trwa 1 miesiąc (30 dni).
- Do zgłoszenia wniosku wymagane jest posiadanie konta i aktywnej afiliacji pracownika naukowego w Portalu PLGrid.

Thank you for your attention!

→ Questions?

◆ pawel.gora@qaif.org

→ WWW:

◆ <http://www.qaif.org>

◆ Resources to learn more: <https://www.qaif.org/resources>

“Logic can get you from A to B, imagination will take you everywhere”
A. Einstein

*“The sky is **NOT** the limit”*

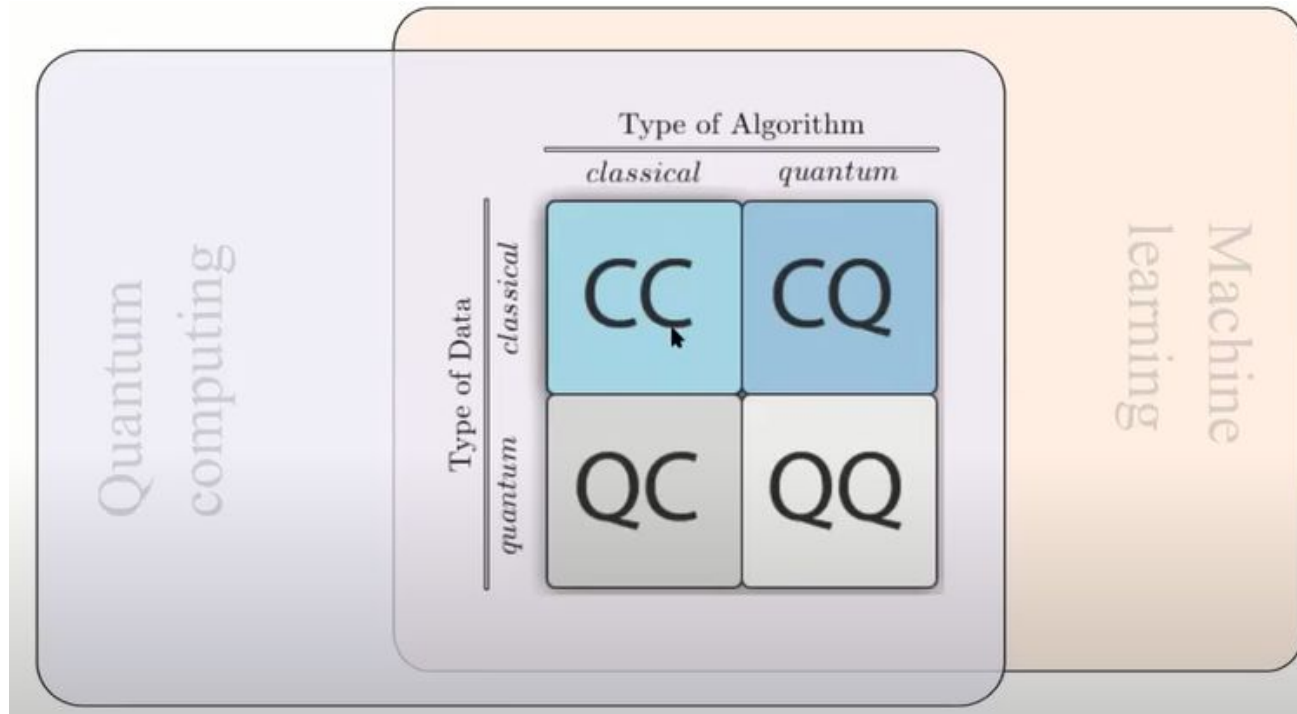


Quantum ML

Training machine learning algorithms using QC (usually with support of classical computers):

- Quantum sampling, Quantum Monte Carlo
- Quantum neural networks
- Calculating average (expected) gradients
- Solving optimization problems (which often occur in AI/machine learning)
- Manipulating matrices (inverse matrix, finding eigenvectors and eigenvalues)
- Ensemble learning
- Distance matrix (k-Means, k-medoids), Quantum kNN
- Quantum SVM
- Quantum Boltzmann Machines
- Quantum GANs
- Quantum RL
- Supervised learning with quantum-enhanced feature spaces
- ...

Quantum ML



Source: “Quantum Machine Learning and PennyLane by Maria Schuld | QWorld”

<https://www.youtube.com/watch?v=pe1d0RyCNxY>

Progress

-
- 2017: D-Wave 2000Q (2048 qubits for special purpose quantum annealing)
- 2017: IBM: 50 qubits (20 available in the cloud)
- 2018: Google Bristlecone: 72 qubits
- 2019: Google's quantum supremacy
- 2020: D-Wave DQM (discrete quadratic model)
- 2020: D-Wave Advantage (> 5000 qubits, new topology of connections)
- 2020: Honeywell QC based on ion traps (128 "quantum volume")
- 2020: IonQ - 32 qubits with low gate errors
- 2021: IBM - 127 qubits
- 2022: Xanadu: 216 qubits (photonic processor)
- 2022: Quantinuum: "quantum volume" = 8192
- 2022: IBM: 433 qubits
- 2023: IBM: 1121 qubits
- 2023: IBM announced quantum supremacy
- 2023: Atom announced quantum computer based on neutral atoms with 1225 qubits
- 2024: IQM: 256 qubits (ion traps) with very high fidelity
- 2024: Google Willow: breakthrough in error correction
- 2025: Microsoft: breakthrough in building topological quantum computers
- 2025: D-Wave announced the first "practical" quantum advantage

Contemporary quantum computers are still noisy, we live in the NISQ (noisy intermediate-scale quantum)

Era: "the leading quantum processors contain about 50 to a few hundred qubits, but are not advanced enough to reach fault-tolerance nor large enough to profit sustainably from quantum supremacy"

Progress



Subscribe

Cutting through the noise

The cover depicts a quantum circuit, representing quantum computing being brought into focus by error mitigation. Quantum computers promise to be substantially faster than their classical counterparts at solving certain problems. But unavoidable environmental noise causes errors in quantum machines, and fixing these errors faster than they accumulate is beyond existing quantum processors. In this week's issue, [Abhinav Kandala and his colleagues](#) show that it is still possible for a quantum computer to outperform a classical computer, by mitigating, rather than correcting, the errors. The researchers used an IBM processor composed of 127 qubits on a chip to generate large, entangled states that simulate the dynamics of spins in a model quantum material and accurately predict properties such as its magnetization. The team also shows that leading classical approximations struggle to produce these results. The researchers suggest that with error mitigation, existing and near-future quantum computers might be good enough to help with problems that are beyond the reach of classical machines. — [show less](#)

Source: <https://www.nature.com/nature/volumes/618/issues/7965>

Building quantum computers

- Ion traps (~256 qubits)
- Superconducting qubits (~1121 qubits)
- Photonic qubits (single photons in an optical circuit) (~216 qubits in the cloud)
- D-Wave's machine (~ 5000 qubits)
- Neutral atoms (~1225 qubits)
- Topological quantum computer (doesn't exist yet)

Future ...

“The best way to predict the future is to create it” A. Lincoln

Obstacles on a road toward quantum supremacy:

- Decoherence (limited quantum state duration)
- Noise (need for error correction and/or error mitigation)
- Need for algorithms and proves: in many cases it is difficult to prove that quantum algorithms are better than classical algorithms (even for the Shor's algorithm)
- Building some quantum gates is still challenging

Quantum Error Correction

“In 1995, Shor followed his factoring algorithm with another stunner: **proof that “quantum error-correcting codes” exist**. The computer scientists Dorit Aharonov and Michael Ben-Or (and other researchers working independently) **proved** a year later that **these codes could theoretically push error rates close to zero**. “This was the central discovery in the ’90s that **convinced people that scalable quantum computing should be possible at all**” said Scott Aaronson, a leading quantum computer scientist at the University of Texas — “that **it is merely a staggering problem of engineering**.”

Source:

<https://www.quantamagazine.org/how-space-and-time-could-be-a-quantum-error-correcting-code-20190103>

<https://venturebeat.com/2019/03/27/ibms-quantum-computation-technique-mitigates-noise-and-improves-accuracy> - IBM's technique

Quantum Error Correction

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

can suffer a bit flip with probability p

$$|\psi\rangle \xrightarrow{p} |\psi'\rangle = \alpha|1\rangle + \beta|0\rangle \quad \rho = (1-p)|\psi\rangle\langle\psi| + p|\psi'\rangle\langle\psi'|$$

Error correction code

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \rightarrow |\psi_C\rangle = \alpha|000\rangle + \beta|111\rangle$$

assume at most one qubit suffers a bit flip

$$|\psi_C\rangle \rightarrow \begin{cases} \alpha|000\rangle + \beta|111\rangle & \text{no error} \\ \alpha|100\rangle + \beta|011\rangle & \text{error on qubit 1} \\ \alpha|010\rangle + \beta|101\rangle & \text{error on qubit 2} \\ \alpha|001\rangle + \beta|110\rangle & \text{error on qubit 3} \end{cases}$$

4 orthogonal subspaces.
Errors can be detected and
corrected without affecting
the coherence

It is difficult to prove that quantum algorithms will be better than classical

QUANTUM COMPUTING

Major Quantum Computing Advance Made Obsolete by Teenager

 58 | 

18-year-old Ewin Tang has proven that classical computers can solve the “recommendation problem” nearly as fast as quantum computers. The result eliminates one of the best examples of quantum speedup.

Source: <https://www.quantamagazine.org/teenager-finds-classical-alternative-to-quantum-recommendation-algorithm-20180731>

Noisy Intermediate Scale Quantum (NISQ)

<https://github.com/quantumlib/cirq> - Cirq - a Python framework for creating, editing, and invoking Noisy Intermediate Scale Quantum (NISQ) circuit.

“We are developing a framework to implement a quantum neural network on near-term processors. We are interested in understanding what advantages may arise from generate massive superposition states during operation of the network.”
(<https://ai.google/research/teams/applied-science/quantum-ai>)

“Quantum supremacy”


Google’s efforts:

“Characterizing quantum supremacy in near-term devices“ (published in “Nature” in 2018):

“Here, we propose the task of sampling from the output distribution of random quantum circuits as a demonstration of quantum supremacy”

Article | Published: 23 October 2019

Quantum supremacy using a programmable superconducting processor

Frank Arute, Kunal Arya, [...] John M. Martinis 

Nature **574**, 505–510(2019) | [Cite this article](#)

684k Accesses | **48** Citations | **6077** Altmetric | [Metrics](#)

Abstract

The promise of quantum computers is that certain computational tasks might be executed exponentially faster on a quantum processor than on a classical processor¹. A fundamental challenge is to build a high-fidelity processor capable of running quantum algorithms in an exponentially large computational space. Here we report the use of a processor with programmable superconducting qubits^{2,3,4,5,6,7} to create quantum states on 53 qubits, corresponding to a computational state-space of dimension 2^{53} (about 10^{16}). Measurements from repeated experiments sample the resulting probability distribution, which we verify using classical simulations. Our Sycamore processor takes about 200 seconds to sample one instance of a quantum circuit a million times—our benchmarks currently indicate that the equivalent task for a state-of-the-art classical supercomputer would take approximately 10,000 years. This dramatic increase in speed compared to all known classical algorithms is an

Source: <https://www.nature.com/articles/s41586-019-1666-5>

Quantum programming frameworks: QISKIT

Run a quantum program

```
[python3] $ pip install qiskit
```

```
from qiskit import QuantumProgram
qp = QuantumProgram()
qr = qp.create_quantum_register('qr', 2)
cr = qp.create_classical_register('cr', 2)
qc = qp.create_circuit('Bell', [qr], [cr])
qc.h(qr[0])
qc.cx(qr[0], qr[1])
qc.measure(qr[0], cr[0])
qc.measure(qr[1], cr[1])
result = qp.execute('Bell')
print(result.get_counts('Bell'))
```

Source: <https://vitalflux.com/quantum-computing-install-qiskit-using-pip>

pyQuil

```
1  # Import pyQuil modules
2  from pyquil.quil import Program
3  from pyquil.api import QVMConnection
4  from pyquil.gates import H
5  from functools import reduce
6
7  # Create a connection to the Quantum Virtual Machine (QVM)
8  qvm = QVMConnection()
9
10 # Apply the Hadamard gate to three qubits to generate 8 possible randomized results
11 dice = Program(H(0), H(1), H(2))
12
13 # 8 possible results: [[0,0,0], [0,0,1], [0,1,1], [1,1,1], [1,1,0], [1,0,0], [0,1,0]] [0,0,1]]
14 # Measure the qubits to get a result, i.e. roll the dice
15 roll_dice = dice.measure_all()
16
17 # Execute the program by running it with the QVM
18 result = qvm.run(roll_dice)
19
20 # Example result: [[0,1,0]]
21 # Format and print the result as a dice value between 1 and 8
22 dice_value = reduce(lambda x, y: 2*x + y, result[0], 0) + 1
23 print("Your quantum dice roll returned:", dice_value)
```

quantum_dice.py

Source: <https://medium.com/rigetti/how-to-write-a-quantum-program-in-10-lines-of-code-for-beginners-540224ac6b45>

ProjectQ

```
from projectq import MainEngine # import the main compiler engine
from projectq.ops import H, Measure # import the operations we want to perform (Hadamard and measurement)

eng = MainEngine() # create a default compiler (the back-end is a simulator)
qubit = eng.allocate_qubit() # allocate a quantum register with 1 qubit

H | qubit # apply a Hadamard gate
Measure | qubit # measure the qubit

eng.flush() # flush all gates (and execute measurements)
print("Measured {}".format(int(qubit))) # converting a qubit to int or bool gives access to the measurement result
```

Source: <https://github.com/ProjectQ-Framework/ProjectQ>

D-Wave Leap

D-WAVE **LEAP**

Take the Leap

Sign up with Leap. Create an account for free time on a D-Wave quantum computer, to learn the basics, and to run your own quantum experiments.

Already have an account? [Log in](#)

COVID-19 project? [Start here](#)

FIRST NAME*

Emily

LAST NAME*

Smith

EMAIL*

emily@gmail.com

I AM A...*

-- Please select a profession --

Source: <https://www.dwavesys.com/take-leap>

Yao.jl

```
0

julia> reg = rand_state(20)
ArrayReg{1, Complex{Float64}, Array...}
  active qubits: 20/20

julia> reg |> subroutine(2-, qft, (4,6,7)) # the this circuit on a sub-register
ERROR: syntax: unexpected ","
Stacktrace:
 [1] top-level scope at REPL[39]:0

julia> reg |> subroutine(20, qft, (4,6,7)) # the this circuit on a sub-register
ArrayReg{1, Complex{Float64}, Array...}
  active qubits: 20/20

julia> # we can also run this simulation on GPU

julia> using CuYao

julia> creg = reg |> cu # upload
ArrayReg{1, Complex{Float64}, CuArray...}
  active qubits: 20/20

julia> creg |> subroutine(20, qft, (4,6,7))
```

PennyLane

(“TensorFlow of quantum computing”)

```
import pennylane as qml
from pennylane import numpy as np

# create a quantum device
dev1 = qml.device('default.qubit', wires=1)

@qml.qnode(dev1)
def circuit(phi1, phi2):
    # a quantum node
    qml.RX(phi1, wires=0)
    qml.RY(phi2, wires=0)
    return qml.expval.PauliZ(0)

def cost(x, y):
    # classical processing
    return np.sin(np.abs(circuit(x, y))) - 1

# calculate the gradient
dcost = qml.grad(cost, argnum=[0, 1])
```

TensorFlow Quantum

```
# A hybrid quantum-classical model.
model = tf.keras.Sequential([
    # Quantum circuit data comes in inside of tensors.
    tf.keras.Input(shape=(), dtype=tf.dtypes.string),

    # Parametrized Quantum Circuit (PQC) provides output
    # data from the input circuits run on a quantum computer.
    tfq.layers.PQC(my_circuit, [cirq.Z(q1), cirq.X(q0)]),

    # Output data from quantum computer passed through model.
    tf.keras.layers.Dense(50)
])
```

Source: <https://www.tensorflow.org/quantum>

QISKIT ML

Run a quantum program

```
[python3] $ pip install qiskit
```

```
from qiskit import QuantumProgram
qp = QuantumProgram()
qr = qp.create_quantum_register('qr', 2)
cr = qp.create_classical_register('cr', 2)
qc = qp.create_circuit('Bell', [qr], [cr])
qc.h(qr[0])
qc.cx(qr[0], qr[1])
qc.measure(qr[0], cr[0])
qc.measure(qr[1], cr[1])
result = qp.execute('Bell')
print(result.get_counts('Bell'))
```

Source: <https://vitalflux.com/quantum-computing-install-qiskit-using-pip>

Qiskit's Machine Learning module (`qiskit.ml`) ¶

This is the Qiskit's machine learning module. There is an initial set of function here that will be built out over time. At present it has sample sets that can be used with Aqua's `classifiers` and circuits used in machine learning applications.

Submodules ¶

`circuit.library`

Circuit library for Machine Learning applications
(`qiskit.ml.circuit.library`)

`datasets`

Datasets (`qiskit.ml.datasets`)

Source: https://qiskit.org/documentation/stable/0.28/apidoc/qiskit_ml.html

QWorld

→ <http://qworld.net>

WHO WE ARE

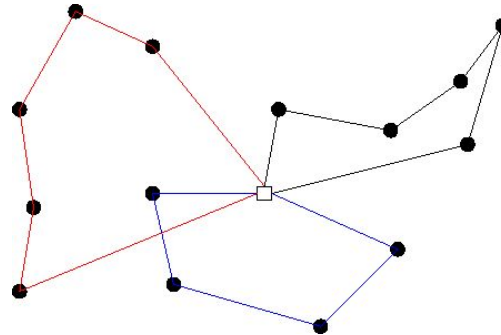
QWorld is a global network of individuals, groups, and communities collaborating on education and implementation of quantum technologies and research activities.

Vehicle Routing Problem and its variants

Input: we have a directed graph with non-negative costs assigned to edges.

Travelling salesman problem (TSP): we have N selected vertices in the graph (customers). What is the shortest possible route that visits all of them and returns to the origin vertex (depot)?

Vehicle Routing Problem (VRP): we have N selected vertices in the graph (customers). What is the optimal (with minimal cost) set of (up to M) routes which (in total) visit all the selected vertices and start and end in the depot?



Vehicle Routing Problem and its variants

Input: we have a directed graph with non-negative costs assigned to edges.

Travelling salesman problem (TSP): we have N selected vertices in the graph (customers). What is the shortest possible route that visits all of them and returns to the origin vertex (depot)?

Vehicle Routing Problem (VRP): we have N selected vertices in the graph (customers). What is the optimal (with minimal cost) set of (up to M) routes which (in total) visit all the selected vertices and start and end in the depot?

Capacitated Vehicle Routing Problem (CVRP): VRP with bounded capacities of vehicles.

Capacitated Vehicle Routing Problem with Time Windows (CVRPTW): VRP with bounded capacities of vehicles and with time windows.

(all these problems are NP-hard)

Solving Vehicle Routing Problem - notation

$T = \{1, 2, \dots, M\}$ - identifiers of trucks/vehicles

$V = \{1, 2, \dots, N, N+1\}$ - identifiers of vertices/nodes ($N+1$ - depot, $1, 2, 3, \dots, N$ - customers)

$C_{i,j}$ - cost of travel from node i to node j (for $i, j \in V$), $C_{i,i} = 0$

$x_{i,j,k} = 1$ if in a given setting the vehicle i visits the node j as k -th location on its route (**0** otherwise)

(i is from the set T , j from V , k is from $\{0, 1, 2, 3, \dots, N+1\}$)

Solving Vehicle Routing Problem - notation

$T = \{1, 2, \dots, M\}$ - identifiers of trucks/vehicles

$V = \{1, 2, \dots, N, N+1\}$ - identifiers of vertices/nodes ($N+1$ - depot, $1, 2, 3, \dots, N$ - customers)

$C_{i,j}$ - cost of travel from node i to node j (for $i, j \in V$), $C_{i,i} = 0$

$x_{i,j,k} = 1$ if in a given setting the vehicle i visits the node j as k -th location on its route (0 otherwise)

(i is from the set T , j from V , k is from $\{0, 1, 2, 3, \dots, N+1\}$)

Observations:

1. For each vehicle i : $x_{i,N+1,0} = 1$, $x_{i,j,0} = 0$ for $j < N + 1$ (the depot is always the initial (0-th) location)
2. If $x_{i,N+1,L} = 1$ for some L , then for $W > L$: $x_{i,j,W} = 0$ for $j < N + 1$ and $x_{i,N+1,W} = 1$ (the depot is always the last location on each route and each car stays in the depot after reaching it)

Solving Vehicle Routing Problem - QUBO formulation

$$C = \sum_{m=1}^M \sum_{n=1}^N x_{m,n,1} C_{N+1,n} + \sum_{m=1}^M \sum_{n=1}^N x_{m,n,N} C_{n,N+1} + \\ + \sum_{m=1}^M \sum_{n=1}^{N-1} \sum_{i=1}^{N+1} \sum_{j=1}^{N+1} x_{m,i,n} x_{m,j,n+1} C_{i,j}$$

- The first component of the sum C is a sum of all costs of travels from the depot to the first visited node - the first section of each car's route.
- The second is a cost of the last section of a route (to depot) in a special case when a single car serves all N orders (only in such a case the component can be greater than 0).
- The last part is the cost of all other sections of routes.

Solving Vehicle Routing Problem - QUBO formulation

Let's consider a binary function:

$$A(y_1, y_2, \dots, y_n) = (y_1 + y_2 + \dots + y_n - 1)^2 - 1$$

where $y_i \in \{0,1\}$ for $i \in \{1, \dots, n\}$. The minimum value of $A(y_1, y_2, \dots, y_n)$ is equal to -1 and this value can be achieved only if exactly one of y_1, y_2, \dots, y_n is equal to 1.

Solving Vehicle Routing Problem - QUBO formulation

To assure that each delivery is served by exactly one vehicle and exactly once and that each vehicle is in exactly one place at a given time, the following term should be included in our QUBO formulation:

$$Q = \sum_{k=1}^N A(x_{1,k,1}, x_{2,k,1}, \dots, x_{1,k,2}, \dots, x_{M,k,N}) + \\ + \sum_{m=1}^M \sum_{n=1}^N A(x_{m,1,n}, x_{m,2,n}, \dots, x_{m,N+1,n})$$

Solving Vehicle Routing Problem - QUBO formulation

Full QUBO Solver (FQS)

By definition of VRP, QUBO representation of this optimization problem is

$$QUBO_{VRP} = A_1 \cdot C + A_2 \cdot Q$$

for some constants A_1 and A_2 , which should be properly set to ensure that the solution found by quantum annealer minimizes the cost C and satisfies the aforementioned constraints (Q).

Solving Vehicle Routing Problem

Average Partition Solver (APS)

We decrease the number of variables for each vehicle by assuming that every vehicle serves approximately the same number of orders - up to $A+L$ deliveries, where A is the total number of orders divided by the number of vehicles (N/M) and L is a parameter (called “limit radius”), which controls the number of orders (in practice, we usually want our fleet to be evenly loaded).

The number of variables is lower which simplifies computations.

Solving Capacitated Vehicle Routing Problem

DBSCAN Solver (DBSS)

Hybrid algorithm which combines quantum approach with a classical algorithm (Recursive DBSCAN).

DBSS uses recursive DBSCAN as a clustering algorithm with limited size of clusters. Then, TSP is solved by FQS separately (we can just assume the number of vehicles equal to 1).

If the number of clusters is equal to (or lower than) the number of vehicles, the answer is known immediately. Otherwise, the solver runs recursively considering clusters as deliveries, so that each cluster contains orders which in the final result are served one after another without leaving the cluster.

We also concluded that by limiting the total sum of weights of deliveries in clusters, this algorithm can solve CVRP if all capacities of vehicles are equal.

Solving Capacitated Vehicle Routing Problem

Solution Partitioning Solver (SPS)

This algorithm divides TSP solution found by another algorithm (e.g., FQS) into consecutive intervals, which are the solution for CVRP.

Let $\mathbf{d}_1, \mathbf{d}_2, \dots, \mathbf{d}_N$ be the TSP solution for \mathbf{N} orders, let \mathbf{P}_v be a capacity of the vehicle \mathbf{v} , let $\mathbf{w}_{i,j}$ be the sum of weights of orders $\mathbf{d}_i, \mathbf{d}_{i+1}, \mathbf{d}_{i+2}, \dots, \mathbf{d}_j$ (in the order corresponding to TSP solution) and let $\mathbf{cost}_{i,j}$ be the total cost of serving only orders $\mathbf{d}_i, \mathbf{d}_{i+1}, \dots, \mathbf{d}_j$. Also, let $\mathbf{dp}_{i,S}$ be the cost of the best solution for orders $\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3, \dots, \mathbf{d}_i$ and for the set of vehicles \mathbf{S} . Now, the dynamic programming formula for solving CVRP is given by:

$$\text{where } \mathbf{cost}_{i,j} = 0 \text{ and } dp_{i,S} = \min_{v \in S, 0 \leq j \leq i, w_{j+1,i} \leq P_v} \{dp_{j,S \setminus \{v\}} + \mathbf{cost}_{j+1,i}\}$$

Solving Capacitated Vehicle Routing Problem

Solution Partitioning Solver (SPS)

To speed up computations, we can apply a heuristic:

1. Instead of set S of vehicles, consider a sequence v_1, v_2, \dots, v_M of vehicles and assume that we attach them to deliveries in such an order.
2. Now, our dynamic programming formula is given by:

$$dp_{i,\{v_1,\dots,v_k\}} = \min_{0 \leq j \leq i, w_{j+1,i} \leq P_{v_k}} \{dp_{j,\{v_1,\dots,v_{k-1}\}} + cost_{j+1,i}\}$$

3. To count this dynamic effectively, we can observe that:

$$\forall j < i, 1 \leq k \leq M (dp_{j-1,v_k} + cost_{j,i}) - (dp_{j-1,v_k} + cost_{j,i-1}) = C_{i-1,i} + C_{i,N+1} - C_{i-1,N+1}$$

We can now select some random permutations of vehicles and perform dynamic programming for each of them.

Experiments - setup

Goal: compare results of different algorithms (quantum, hybrid, and classical) on several datasets.

For quantum/hybrid algorithms, we ran experiments using **D-Wave's Leap platform** and 2 solvers: **qbsolv** (on QPU or CPU) and **hybrid solver** (on QPU and CPU at the same time).

In case of classical algorithms, we tested Simulated Annealing (SA), Bee Algorithm (BEE), Evolutionary Annealing (EA), DBSCAN with simulated annealing (DBSA).

Experiments - datasets

We prepared several datasets:

- Christofides1979 - a standard benchmark dataset for CVRP, well-known and frequently studied by the scientific community (14 tests with different number of vehicles, capacities and number of orders).
- A dataset built by us based on a realistic road network of Belgium, acquired from the OpenStreetMap service (51 tests).

Experiments - datasets

Christofides1979 - a standard benchmark dataset for CVRP, well-known and frequently studied by the scientific community (14 tests with different number of vehicles, capacities and number of orders).

Test name	Nr of vehicles	Capacity	Nr of orders
CMT11	7	200	120
CMT12	10	200	100
CMT13	11	200	120
CMT14	11	200	100
CMT3	8	200	100
CMT6	6	160	50
CMT7	11	140	75
CMT8	9	200	100
CMT9	14	200	150

Parameters of instances of Christofides1979 used in our experiments.

Experiments - datasets

A dataset built by us based on a realistic road network of Belgium, acquired from the OpenStreetMap service (51 tests): we considered different numbers of orders (from 1 to 200) and different locations of orders and depots.

Test	Number of orders
small-0	2
small-1	2
small-2	2
small-3	1
small-4	2
small-5	5
small-6	6
small-7	5
small-8	4
small-9	6
medium-0	20
medium-1	26
medium-2	27
medium-3	24

Examples of test cases from realistic road networks

Results of experiments

Test	vehicles	FQS CPU	FQS QPU	FQS Hybrid	APS CPU	APS Hybrid
small-0	1,2	11286	11286	11286	11286	11286
small-1	1	10643	10643	10643	10643	10643
	2	10643	10643	10643	12379	12379
	3	10643	-	10643	-	-
small-2	1	21311	21311	21311	21311	21311
	2	21311	-	21311	24508	24508
	3	22192	-	21311	-	-
small-3	1	18044	18044	18044	18044	18044
	2	20819	-	18033	22193	22193
	3	22843	-	18033	-	-
small-4	1	15424	15424	15424	15424	15424
	2	17364	-	15424	19472	19472
	3	17364	-	15424	-	-
small-5	1	10906	10906	10906	10906	10906
	2	11676	-	10906	13480	13480
	3	11754	-	10906	-	-
small-6	1	20859	20859	20859	20859	20859
	2	26735	-	20859	26735	26735
	3	27110	-	20859	-	-

Results for different quantum and hybrid algorithms on some small datasets

Results of experiments

Test	vehicles	FQS CPU	FQS QPU	FQS Hybrid	APS CPU	APS Hybrid	DBSS CPU
medium-0	1	20774	-	21775	20774	21775	24583
	2	36966	-	29879	25737	25217	27994
	3				28226	27237	34185
medium-1	1	29868	-	29423	29868	29423	27606
	2	50639	-	39485	30820	31129	31346
	3	-	-	-	33376	32018	32588
medium-2	1	37045	-	35208	37045	35208	29442
	2	55579	-	36511	33235	33163	32947
	3	-	-	-	36600	32569	34480
medium-3	1	30206	-	29422	30206	29422	31092
	2	51787	-	35774	31428	30273	33790
	3	-	-	-	35994	33627	33712
medium-4	1	21257	-	20762	21257	20762	21435
	2	34379	-	25470	22410	22722	22885
	3	-	-	-	23599	22176	25446
medium-5	1	23013	-	21642	23013	21462	21737
	2	36149	-	22041	22775	23076	23403
	3	-	-	-	24899	22386	24336
medium-6	1	23804	-	24664	23804	23804	23926
	2	35826	-	24490	24265	25178	25510
	3	-	-	-	27032	23364	25122

Results for different quantum and hybrid algorithms on some medium datasets

Results of experiments

	vehicles	APS CPU	DBSS CPU
big-0	1	80084	71594
	2	97286	71051
big-1	1	157660	146828
	2	206782	149200
big-2	1	168646	154105
big-3	1	85873	62236
big-4	1	156411	129279

Comparison of results for Average Partition Solver and DBSCAN Solver on big test cases.

Results of experiments

	vehicles	capacity	SPS (CPU)	DBSS (CPU)
big-0	2	100	70928	73508
	2	85	72295	73189
	2	80	75150	Not valid
	3	100	71320	76717
	3	70	71251	78012
	3	55	Not valid	76807
	5	100	71740	Not valid
	5	50	78726	91066
	5	40	85976	Not valid
big-1	2	100	150608	158631
	2	80	150608	152946
	2	65	150804	156188
	3	100	151525	153673
	3	60	153190	152854
	3	45	164055	Not valid
	5	100	151930	168789
	5	40	156242	165271
	5	30	174519	176935

Comparison of DBSCAN Solver and Solution Partitioning Solver (SPS) run on CPU on big test cases with various capacities.

Results of experiments

	type	deliveries	SPS	Simul. Ann.	Bee	Evolution
clustered1-1	average	57	69850	66379	60876	48923
	best	57	69080	52119	56358	48152
clustered1-2	average	55	77173	74341	81438	54719
	best	55	75530	59947	68772	53490
group1-1	average	42	158919	156217	153495	137989
	best	42	155388	146526	142774	135593
group1-2	average	54	171732	145380	145325	137626
	best	54	165043	141065	140947	136307
range-6-1	average	47	71670	68003	67234	59937
	best	47	68459	62312	64404	59827
range-6-2	average	50	80490	84380	83915	73651
	best	50	79640	79574	85917	73051
range-8-12-1	average	50	142008	146553	142835	129069
	best	50	140170	136369	127372	126555
range-8-12-2	average	50	146798	137628	145332	129048
	best	50	143598	135493	136776	128803
range-8-12-3	average	46	105544	105051	98366	92792
	best	46	101577	99004	94423	91921
range-8-12-4	average	51	147993	143309	148900	128316
	best	51	145559	140088	128575	124405
range-8-12-5	average	50	146719	143516	145685	134162
	best	50	143993	139784	139796	133245
range-8-12-6	average	50	146984	148194	150121	136326
	best	50	141467	138781	139400	134692
range-5-1	average	50	81728	68900	69052	67896
	best	50	72527	67984	68022	67691
range-5-2	average	50	81759	69342	68564	67981
	best	50	76868	67958	67780	67716

Results of Solution Partitioning Solver compared with results for classical algorithms run on artificially generated test cases.

Results of experiments

Test name	SPS	SA	BEE	EA	DBSA
CMT11	25.54	23.62	36.18	16.52	19.94
CMT12	26.84	53.06	20.24	20.68	21.37
CMT13	25.97	86.72	34.66	35.05	19.44
CMT14	26.83	52.52	20.23	20.23	22.8
CMT3	25.13	48.3	28.38	28.82	-
CMT6	17.58	48.3	15.42	28.82	15.82
CMT7	29.42	41.4	27.89	31.68	23.18
CMT8	26.5	51.16	26.67	28.09	19.4
CMT9	34.14	76.34	44.25	42.81	-

Comparison of results achieved by Solution Partitioning Solver (SPS) and classical algorithms (SA - simulated annealing, BEE- Bee algorithm, EA - evolutionary annealing, DBSA - DBSCAN with simulated annealing) on a benchmark dataset Christofides79.

Conclusions

- It doesn't make sense to run experiments on QPU for large test cases
- SPS (with qbsolv run on CPU) gives the best results among “quantum” algorithms
- We compared SPS with some classical metaheuristics for CVRP well-known in the scientific literature (e.g., simulated annealing (SA), bee algorithm (BEE), evolutionary annealing (EA), DBSCAN with simulated annealing (DBSA)) - it usually gives a bit worse (but sometimes better) results.
- Github repository: <https://github.com/dwave-examples/D-Wave-VRP>

Other research directions

- VQE, QAOA and their variants (limited power due to small number of qubits)
- Scalability - graph clustering / coarsening
- Investigate for which scenarios the hybrid algorithms give the best results comparing to classical algorithms (for which scenarios we may expect some advantages?)
- CVRPTW and other variants
- Applying machine learning techniques to good values of the relaxation parameter
- Other QUBO formulations

Acknowledgement

- The presented research is based on the article **“New hybrid quantum annealing algorithms for solving Vehicle Routing Problem”** and was presented at the **ICCS 2020** conference.
- **Michał Borowski, Paweł Gora, Katarzyna Karnas, Mateusz Błajda, Krystian Król, Artur Matyjasek, Damian Burczyk, Miron Szewczyk, Michał Kutwin**, (2020) New Hybrid Quantum Annealing Algorithms for Solving Vehicle Routing Problem. In: Krzhizhanovskaya V. et al. (eds) Computational Science – ICCS 2020. ICCS 2020. Lecture Notes in Computer Science, vol 12142. Springer, Cham. https://doi.org/10.1007/978-3-030-50433-5_42
- https://link.springer.com/chapter/10.1007/978-3-030-50433-5_42

Acknowledgement

The presented research was carried out within the frame of the project “Green LAst-mile Delivery” (GLAD) realized at the University of Warsaw with the project partners: Colruyt Group, University of Cambridge and Technion. The project is supported by EIT Food, which is a Knowledge and Innovation Community (KIC) established by the European Institute for Innovation & Technology (EIT), an independent EU body set up in 2008 to drive innovation and entrepreneurship across Europe.



EIT Food is supported by the EIT
a body of the European Union