Quantum Algorithms

Table of contents

1. Introduction

Note
The reader must have reasonable knowledge of mathematics, physics and classical computing, as well as basic knowledge of quantum computing (as for example included in the articles 'Quantum Computing Explained', 'Quantum Annealing Explained' and 'Quantum Software Development Tools' published by the NOREA Taskforce Quantum Computing).

The first quantum algorithm was already developed in 1992 by the British physicist David Elisier Deutsch and the Australian mathematician Richard Jozsa at a time when quantum computers were not yet available (Figure 1.1). Since then, quantum hardware has caught up with quantum algorithms, but even today, many of the quantum algorithms that are invented are not yet executable on a large problem scale on current quantum computers or on classical computing quantum emulators.

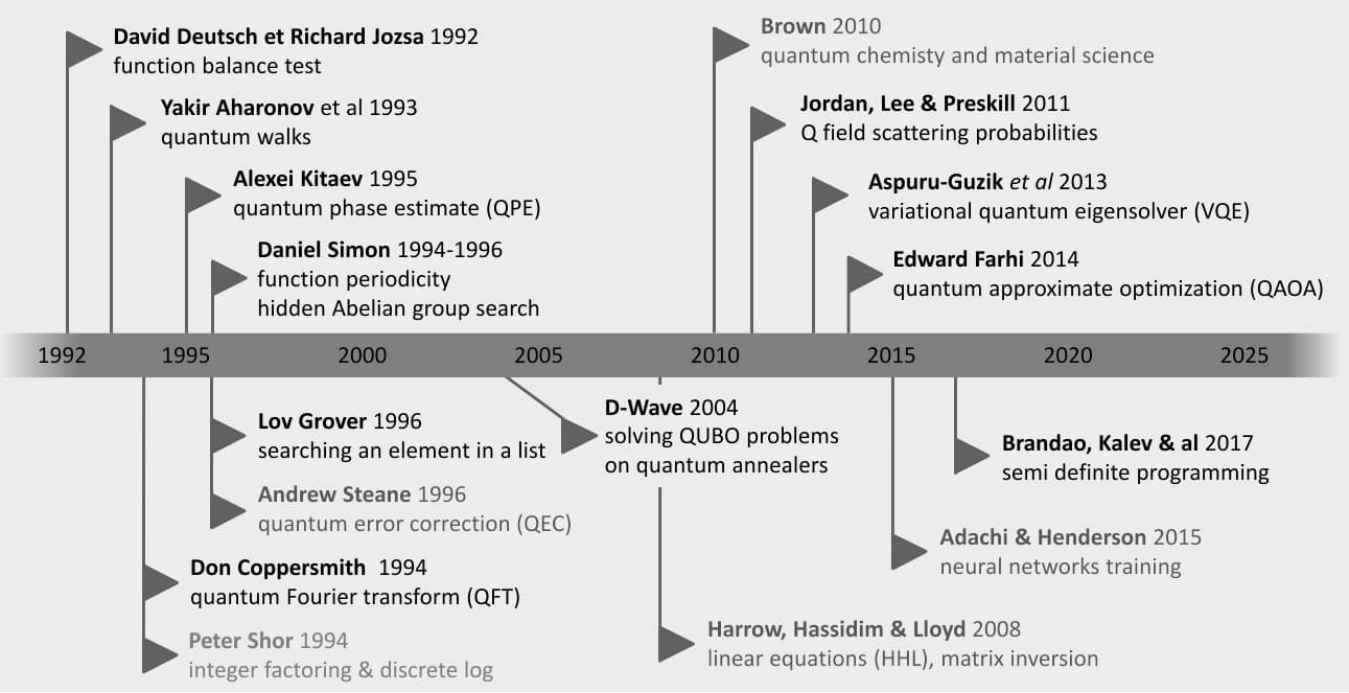

Figure 1.1: Quantum algorithm timeline (source: Olivier Ezratty 2024)

The Deutsch-Jozsa quantum algorithm and Simon's quantum algorithm (developed in 1994 by the American computing security scientist Daniel R. Simon) demonstrated that quantum computers could solve specific problems exponentially faster than classical computers.

Quantum computing has the potential to solve some types of complex (aka "exponential" aka "hard") problems that are currently intractable for classical computers. With its fundamental principles rooted in quantum mechanics, this technology opens up new possibilities for computational power and algorithmic advancements.

Quantum computing use cases fall in three main categories (Figure 1.2):

1. fundamental research;
2. applied research;
3. business operations.

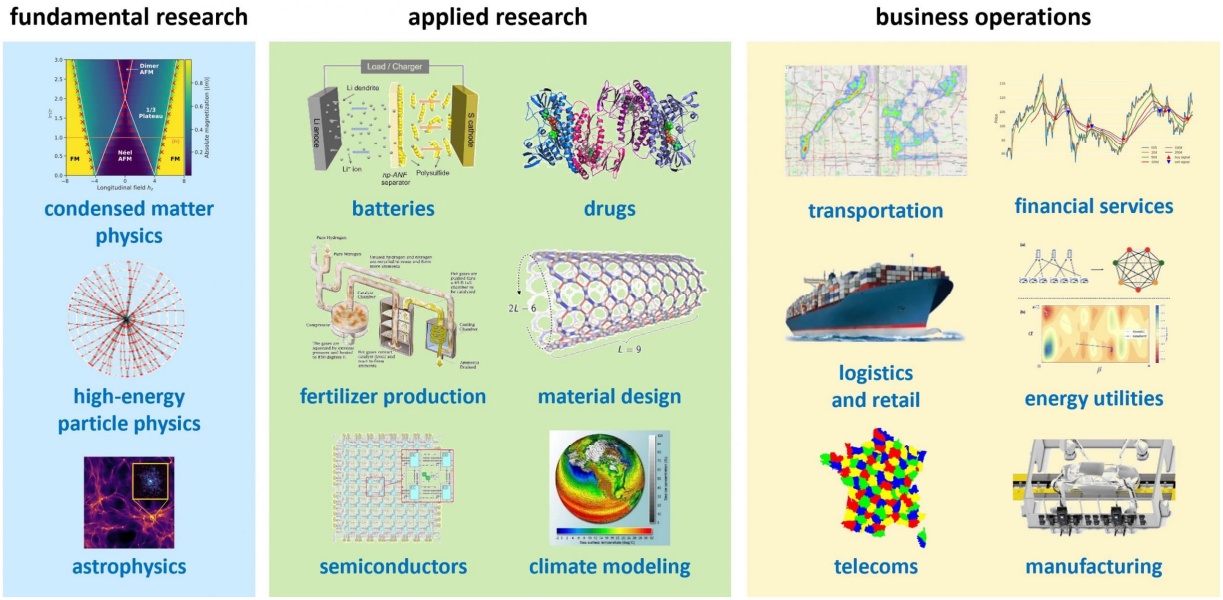

Figure 1.2: Quantum computing use cases (source: Olivier Ezratty 2024)

In general, three important types of problems are well-suited for solving with quantum computing: simulation, optimisation and machine learning. (Generic) quantum algorithms have been conceived for solving these problems.

When developing quantum applications, the problem to be solved by the use of quantum computing must be determined first. After the problem has been analysed and properly understood, the next step is to design the quantum algorithm and then design and specify the corresponding quantum circuit, i.e. the set of qubits, the sequence of operations (quantum gates) to be performed on these qubits, and the qubit measurements yielding the (classical) outcome of the quantum computation.

Qubits can be in state $|0\rangle$, in state $|1\rangle$ or in a linear combination (superposition) $\alpha|0\rangle + \beta|1\rangle$ of both states. The amplitudes α and β correspond with the probabilities that their measured value is either “0” or “1”. The trick in devising an algorithm for a quantum computer is to choreograph a pattern of constructive and destructive interference for its qubits, so that for each wrong answer the contributions to these qubit amplitudes cancel each other out, whereas for the right answer the contributions reinforce each other. If, and only if, that can be arranged, the right answer will be obtained with a large probability when reading the quantum computer’s qubits. The difficulty

is to do this without knowing the answer in advance and, of course, significantly faster than could be done with a classical computer.

Quantum computing is inherently probabilistic, requiring executing several times a quantum calculation and averaging the obtained results. One run of a quantum algorithm is probabilistic but by running the quantum algorithm many times, progressive convergence to a deterministic solution will be achieved (the solution being the average of all run results). The number of runs needed is typically in the order of hundreds or thousands. Experience shows that this number will grow with the number of qubits used (hopefully only linearly).

2. Quantum algorithm taxonomy

Note
This article describes only quantum algorithms that can be used as building blocks for quantum algorithms that support specific quantum business applications.

2.1. Low-level quantum algorithms

Today not that many quantum algorithms have been developed compared with the enormous amount of existing classical computing algorithms. Furthermore, most high-level quantum algorithms that have been developed are based on a relatively small set of low-level quantum algorithms and related techniques (Figure 2.1).

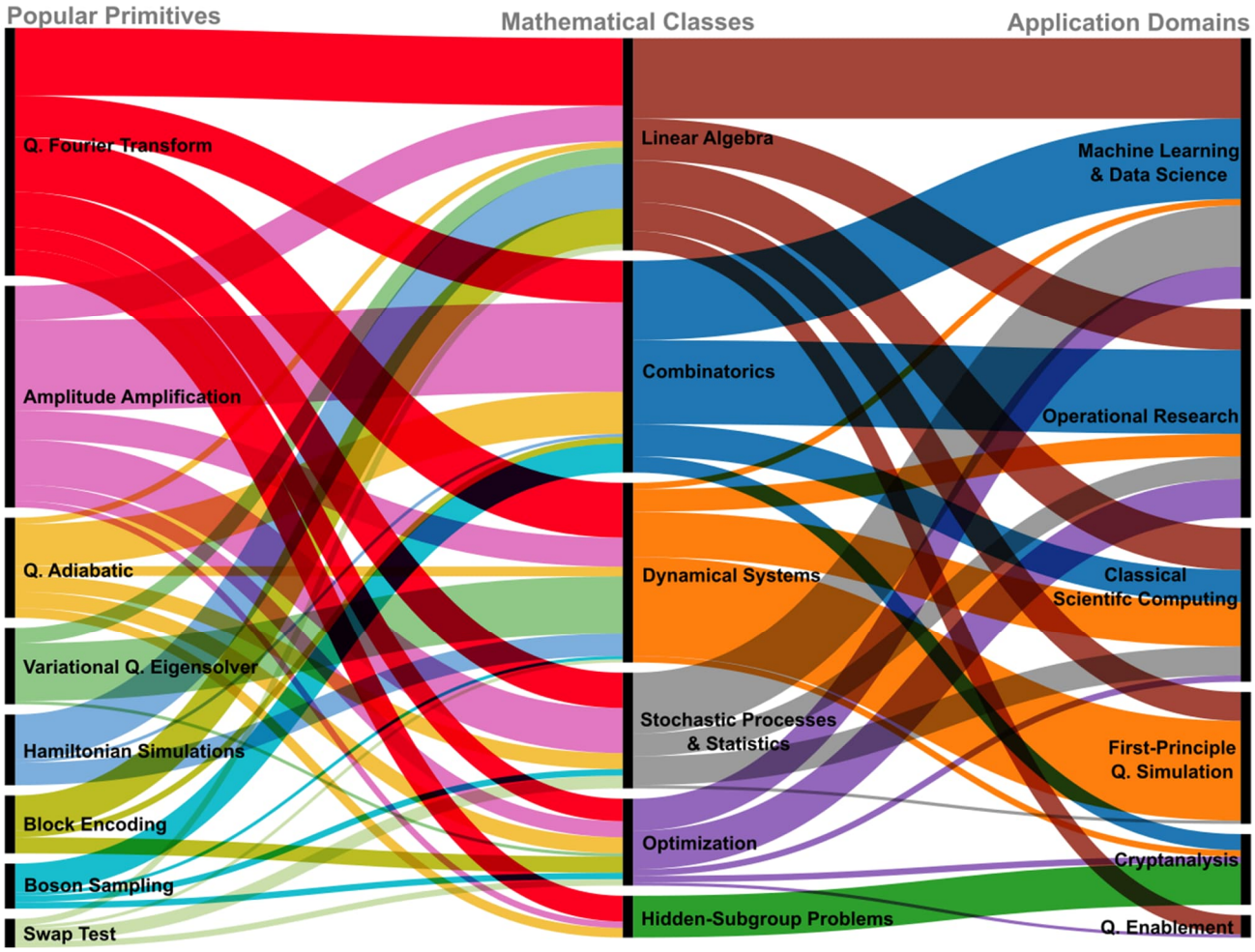

Figure 2.1: Use of quantum algorithm primitives (source: Pablo Arnault et al. 2024)

The quantum algorithm toolbox (Figure 2.2) includes the following well-known elementary quantum algorithms.

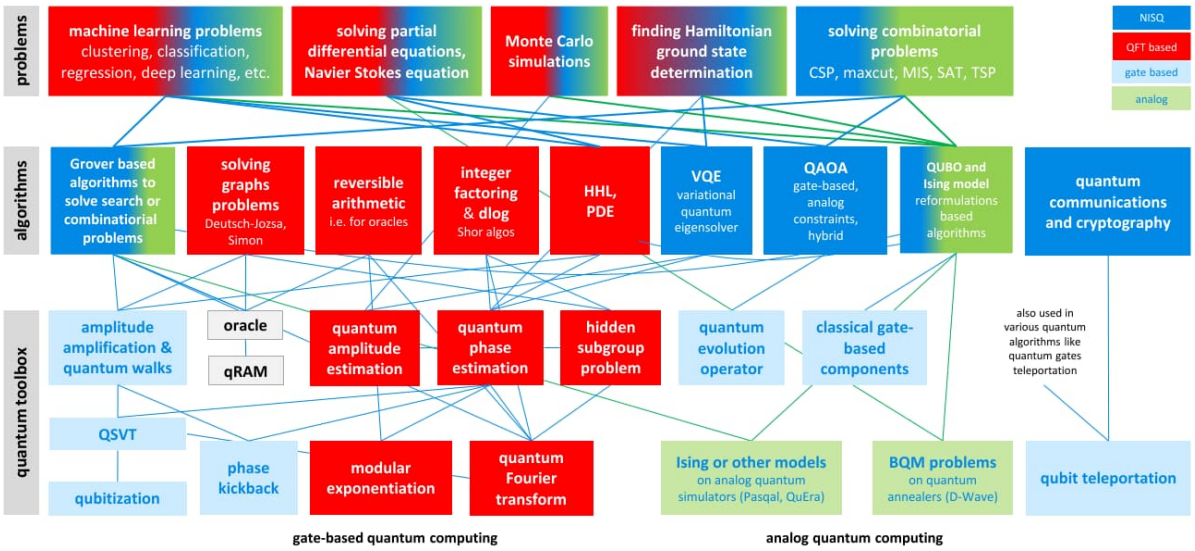

Figure 2.2: Quantum algorithm toolbox (source: Olivier Ezratty 2024)

Grover's algorithm

Grover's algorithm aka quantum search algorithm, invented by the Indian-American computer scientist Lov Kumar Grover, is a quantum algorithm for unstructured search that finds with high probability the unique input to a black box function that produces a particular output value. Given its quantum circuit requirements, implementation of Grover's algorithm requires an FTQC quantum computer (Box 2.1) to be useful, i.e. to solve search problems of practical problem sizes.

Qubits are vulnerable to perturbances caused by the environment in which they operate, causing decoherence of the qubit's quantum state. Quantum Error Correction (QEC) is seen as the solution to the qubit decoherence problem. QEC enables sets of noisy physical qubits (imperfect qubits) to emulate stable logical qubits (perfect qubits) so that the quantum computer behaves reliably. Fault-Tolerant Quantum Computers (FTQCs) are quantum computers made more robust through deployment of QEC and other fault reduction techniques (e.g. reliable error correction, reliable qubit readout, etc.).

Box 2.1: Fault-Tolerant Quantum Computer (FTQC)

Quantum Fourier Transform (QFT)

QFT, invented by the American mathematician and computer scientist Don Coppersmith in 1994, is the quantum equivalent of Discrete Fourier Transform (DFT) or Fast Fourier Transform (FFT) classical Fourier transform (Box 2.2) algorithm. An inverse QFT is a QFT executed backwards, with its quantum gates serialised in reverse order. Many other quantum algorithms are using QFT,

including QPE and HHL. To be useful, QFT requires a FTQC quantum computer with a large number of low-error qubits.

> The Fourier transform is a mathematical decomposition of a time domain signal into elementary single frequency signals with their frequency, amplitude and phase. It is a complex value function of time with, for each frequency, a magnitude (real part) and a phase offset (complex part) of the sinusoid of this elementary frequency. The inverse Fourier transforms that frequency decomposition function back into its original compound signal. Fourier series were invented by the French mathematician and physicist Jean-Baptiste Joseph Fourier.

Box 2.2: Fourier transform

Quantum Phase Estimation (QPE)

QPE, invented in 1995 by the Russian-American theoretical physicist Alexei Yurievich Kitaev, is based on QFT and modular exponentiation to find the eigenvalues (Box 2.3) or eigenvalues' phase of a unitary matrix or quantum sub-circuit. It is used in many other linear algebra quantum algorithms.

> A system's set of energy eigenvalues, or its energy spectrum, is the set of possible outcomes obtainable from a measurement of the system's total energy. The word "eigenvalue" is derived from the German word "eigen", meaning "inherent" or "characteristic".

Box 2.3: Eigenvalue

Shor's algorithms

Based on QFT/QPE, the American mathematician Peter Williston Shor invented the period-finding quantum algorithm and the quantum algorithms for solving the factoring and discrete logarithm (dlog) problems, both of which are instances of the period-finding algorithm.

Quantum Amplitude Estimation (QAE)

The QAE quantum algorithm is used to amplify and select the desired state, i.e. the amplitudes, of a quantum superposition. It is used for solving combinatorial search and optimisation problems like the TSP problem (Box 2.4).

> The Travelling Salesperson Problem (TSP) is an optimisation problem where a salesperson must visit a given set of cities exactly once, starting and ending at the same city. The goal is to find the shortest possible route that covers all the cities and returns to the starting point.

Box 2.4: TSP problem

Quantum Amplitude Amplification (QAA)

The QAA quantum algorithm is used to change the probability distribution modelled by a quantum state by increasing the probability of measurement of so-called marked items. It was created by Lov Grover to improve his algorithm for unstructured search.

Quantum numerical solvers

Quantum algorithms have been developed for so-called “numerical solvers”. This includes linear equation (Box 2.5) quantum solvers such as the HHL quantum algorithm (Box 2.6) and other numerical quantum solvers, such as the Partial Differential Equation (PDE) solving quantum algorithm (Box 2.7).

A linear equation is of the form $a_1x_1 + a_2x_2 + \ldots a_nx_n + b$, where x_i are the variables (unknowns), and b and a_i are the coefficients. Linear equations are frequently used in physics, partly because non-linear systems are often well approximated by linear equations.

Box 2.5: Linear equation

The Harrow, Hassidim and Lloyd (HHL) quantum algorithm, named after the American physicist Aram Wettroth Harrow, the Israeli computer scientist Avinatan Hassidim and the American mathematician and philosopher Seth Lloyd, calculates the inverse of a large matrix.

Box 2.6: HHL quantum algorithm

A partial derivative of a function of several variables is its derivative with respect to one of those variables, with the others held constant (as opposed to the total derivative, in which all variables are allowed to vary). Partial derivatives are for example used in vector calculus.

Box 2.7: Partial derivative

Binary Quadratic Model (BQM)

The BQM model defines an objective function (Box 2.8) that is to be optimised with binary variables, a quadratic component and linear constraints.

An objective function is either a cost function (aka loss function) or a profit function (aka reward function), which an optimisation problem seeks to minimise (cost function) or maximise (profit function).

Box 2.8: Objective function

Many combinatorial and optimisation problems such for example the Ising model aka Lenz-Ising model (named after the German physicists Ernst Ising and Wilhelm Lenz) and Quadratic Unconstrained Binary Optimization (QUBO) problem can be translated or converted into BQM.

Hamiltonian simulation

Hamiltonian (Box 2.9) simulation is used to find a point of equilibrium of a complex system such as in quantum physics simulation, neural networks training, searching for optimal paths in networks or process optimisation.

The Hamiltonian of a quantum system, named after the Irish mathematician and physicist William Rowan Hamilton, is an operator corresponding to the total energy of that system, including both kinetic energy and potential energy.

Box 2.9: Hamiltonian

Hamiltonian simulation can be implemented in all quantum paradigms: quantum annealing, analogue quantum computing and universal gate-based quantum computing. In the latter case, this is commonly done with a QPE quantum algorithm on a FTQC quantum computer (Box 2.2) or with a VQE quantum algorithm on a NISQ quantum computer (see Chapter 3).

Trotterization, aka Trotter-Suzuki decomposition, named after the Canadian-American mathematician Hale Freeman Trotter and the Japanese physicist Masuo Suzuki, is a classical computing method that decomposes the unitary time operator of quantum dynamics simulation in small discrete steps. The method lends itself naturally to developing a similar decomposition on quantum computers: the Trotterization quantum algorithm. This quantum algorithm is currently thought to be unfeasible for long time evolution quantum dynamics simulation on a NISQ quantum computer because the complexity of the quantum circuit grows exponentially with the size of the quantum system to be simulated and the width and depth of the quantum circuit (Box 2.10) grows linearly with the time span.

The quantum circuit width is the number of qubits (some quantum circuits are narrow while others are wide); the quantum circuit depth is the number of quantum gates (some quantum circuits are shallow while others are deep). This is a bit confusing because in most graphical representations of quantum circuits, the qubits are shown from top to bottom and the quantum gates are shown from left to right.

Box 2.10: Quantum circuit width and depth

2.2. Some examples of high-level quantum algorithms

Several different if not inconsistent quantum algorithm classification schemes are used in the available literature. The following paragraphs provide a brief overview of some high-level quantum algorithms in each of the following categories:

- oracular;
- quantum simulation;
- quantum optimisation;
- quantum machine learning.

Oracular

Oracular quantum algorithms, aka oracle-based quantum algorithms, include Grover's search algorithm and many variants to solve combinatorial optimisation problems. Examples: TSP problem (Box 2.4), MaxCut problem (Box 2.11) and quantum walks (Box 2.12).

The Maximum Cut (MaxCut) problem is an optimisation problem in which the nodes of a given undirected graph have to be divided in two sets such that the number and weight of edges connecting nodes of the same type are maximised.

Box 2.11: Maximum Cut problem

A random walk is a random process that describes a path that consists of a succession of random steps in some mathematical space. Quantum walks are quantum analogues of classical random walks. In contrast to classical random walks, where the walker occupies definite states and the randomness arises due to stochastic transitions between states, in quantum walks randomness arises through either quantum superposition of states, non-random, reversible unitary evolution of the quantum system or collapse of the quantum wave function due to quantum state measurements. A quantum algorithm for solving quantum walks was invented in 1993 by Yakir Aharonov (Israeli physicist) et al.

Box 2.12: Quantum walks

Oracular quantum speedup claims nearly never take into account the potential computing overhead imposed by the oracle itself. In an ideal world, oracle implementation complexity should scale linearly or at worst polynomially with the number of handled qubits. But the practical oracle scaling overhead could be highly detrimental to any potential theoretical quantum speedup.

Quantum simulation

Quantum simulation algorithms have many and widely different applications (Figure 2.3). They are for example used to simulate the interaction between atoms in molecules for the creation of new materials and they can simulate physical phenomena related to magnetism or the interaction between photons and matter (this amounts to solving "N-body problems", i.e. calculating the interaction between several particles according to the physical laws governing their interaction).

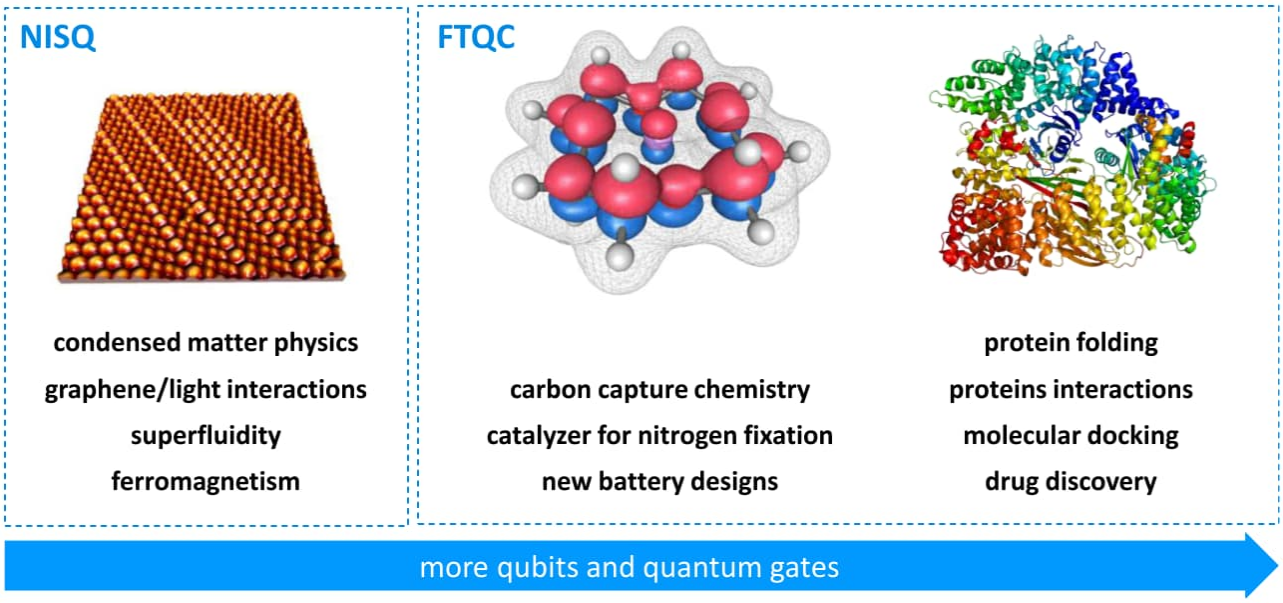

Figure 2.3: Quantum simulation applications (source: Olivier Ezratty 2024)

Quantum optimisation

Quantum optimisation algorithms (Figure 2.4) usually solve a decision problem, which consists in determining (i.e. deciding) whether there exists a solution to the given problem.

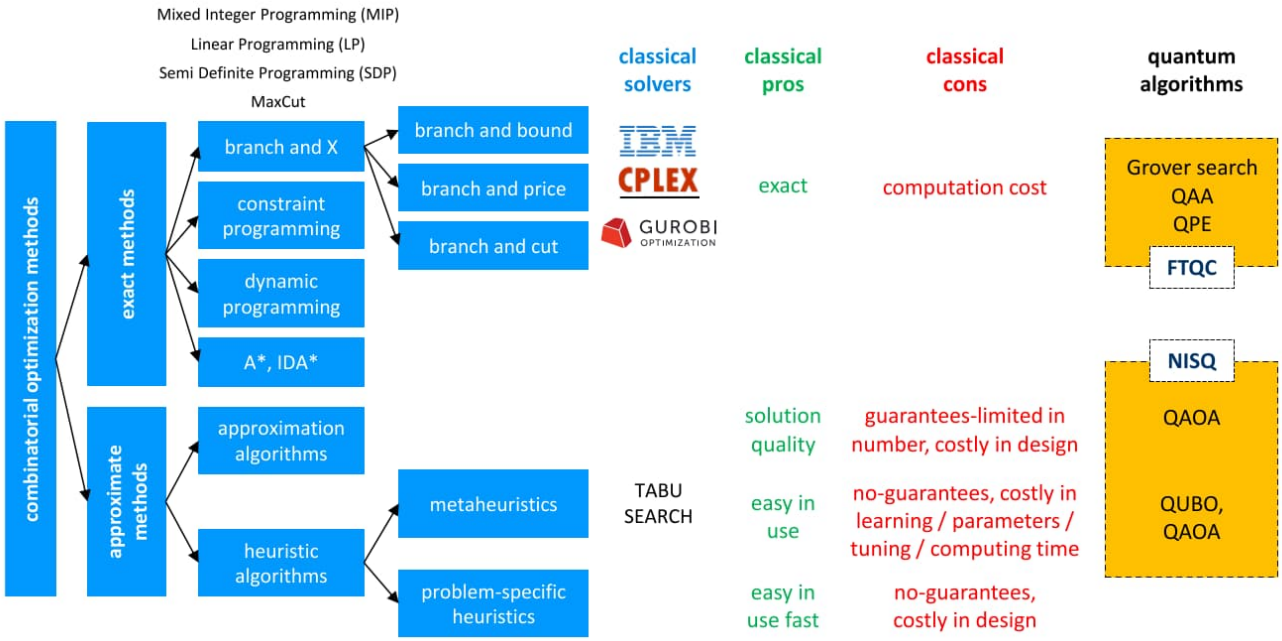

Figure 2.4: Comparison of optimisation methods (source: Olivier Ezratty 2024)

Obtaining efficient optimisation algorithms has become the focus of much research interest since current developing trends in machine learning and cutting-edge applications require complex

optimised models containing a huge number of parameters. At present, classical computers are inefficient for solving many of such complex and wicked optimisation problems. Quantum computers are seen as the solution but this technology is currently still at an early stage.

Quantum-Inspired Algorithms (QIAs) have emerged trying to fill the gap between the theoretical and real quantum computing capabilities. QIAs use classical computers to simulate some quantum phenomena such as superposition and entanglement in order to perform simulated quantum computations. Notable QIA examples are:

- Simulated Quantum Annealing (SQA): SQA outperforms classical simulated annealing for certain problems. It represents a new classical computational strategy that emulates quantum annealing dynamics.

- QIAs for linear algebra, for example: quantum-inspired recommendation system algorithms, quantum-inspired Principal Component Analysis (qPCA) algorithm (Box 2.13), quantum-inspired supervised clustering algorithm, quantum-inspired low-rank stochastic regression algorithm and quantum-inspired sublinear algorithm for solving low-rank linear systems.

The qPCA quantum algorithm is a quantum-enhanced version of the classical Principal Component Analysis (PCA) algorithm, designed to work on quantum datasets. It identifies principal components as quantum states, leveraging quantum computing to efficiently process high-dimensional data and potentially offer speedups over classical methods. qPCA is crucial for dimensionality reduction in quantum datasets, preserving essential data structure while reducing features. It has applications in quantum machine learning, data compression and visualisation, offering benefits in handling complex quantum correlations and uncovering hidden patterns in quantum information.

Box 2.13: qPCA quantum algorithm

Quantum machine learning

Various quantum algorithms have been created in the last decades that cover the field of machine learning, with a lot of variations in neural networks and deep learning. Quantum Machine Learning (QML) algorithms, which are considered a subset of QAI tools (Box 2.14 and Figure 2.5), are either targeting NISQ platforms with variational quantum algorithms or FTQC platforms with linear algebra-based algorithms.

Quantum Artificial Intelligence (QAI) is a very broad computational field that contains QML, quantum reasoning, Quantum Automated Planning and Scheduling (QPS), Quantum Natural Language Processing (QNLP), Quantum Computer Vision (QCV), and Quantum Multi-Agent Systems (QMAS). Note that solving decision problems also belongs to the field of QAI but is not necessarily implemented with QML techniques.

Box 2.14: Quantum Artificial Intelligence (QAI)

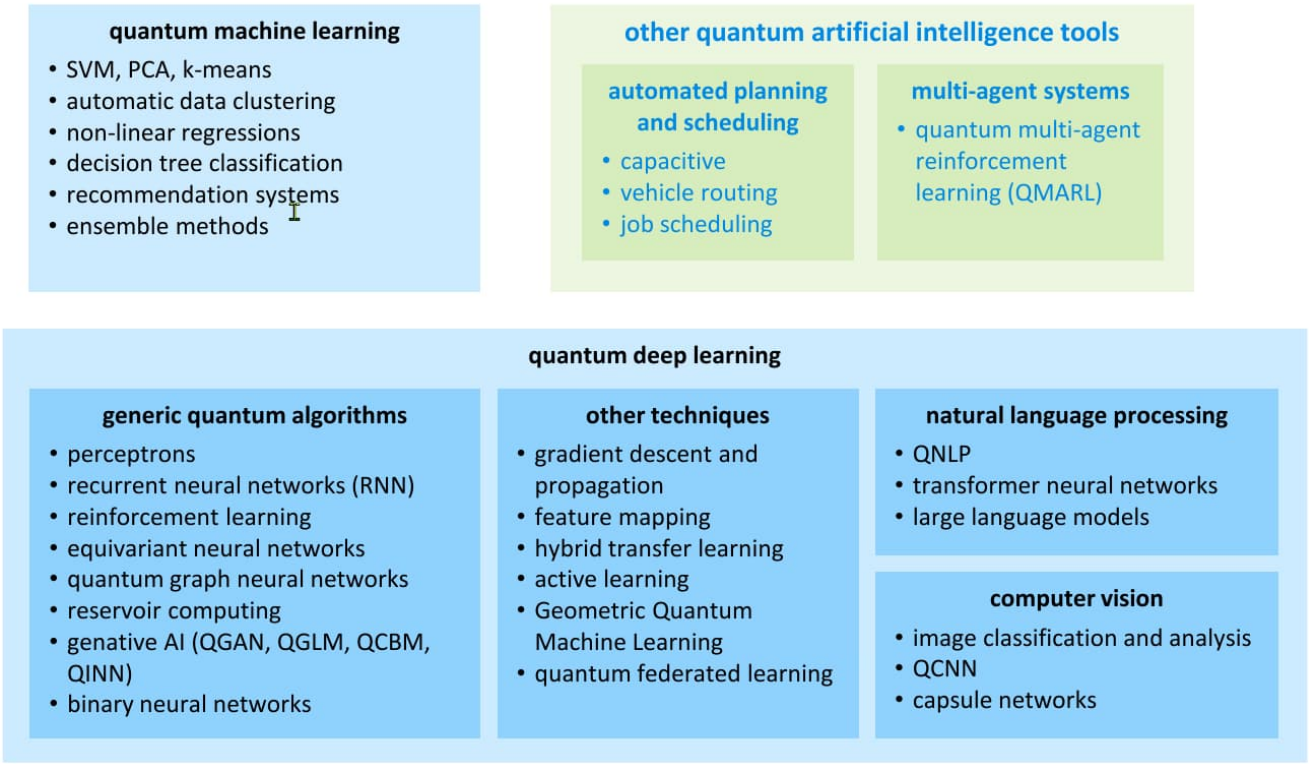

Figure 2.5: QAI tools **(source: Olivier Ezratty 2024)**

The four models defined for QML (Figure 2.6) are:

1. Classical-Classical (CC): classical data that is processed by classical algorithms; this is classical Machine Learning (ML).

2. Classical-Quantum (CQ): classical data that is encoded in quantum states and processed by Variational Quantum Algorithms (VQAs), which combine a quantum algorithm and a classical algorithm that drives it (see Chapter 3). Such VQAs may need the use of a quantum RAM (qRAM).

3. Quantum-Classical (QC): quantum data that is converted in a classical form and processed by classical algorithms; they are used to analyse quantum physics and for obtaining quantum sensor measurement statistics, e.g. for qubit tomography.

4. Quantum-Quantum (QQ): quantum data that is processed by quantum algorithms which could be implemented by feeding a QML algorithm directly with quantum data coming from a quantum sensor or another Quantum Processing Unit (QPU).

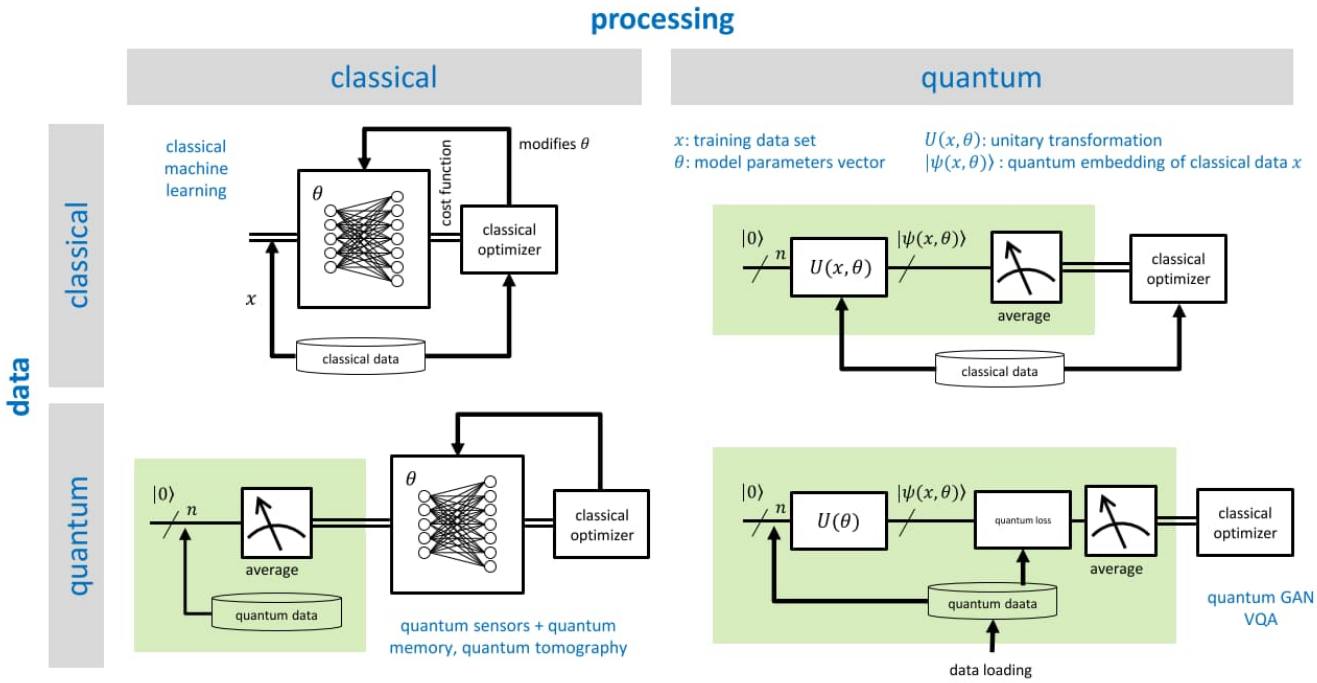

Figure 2.6: The four models of QML (source: Olivier Ezratty 2024)

Several challenges remain to be addressed to operationalise QML on future FTQC quantum computers:

- Loading training data may take time and has a negative impact on the acceleration provided by QML. It also requires qRAM which does not yet exist.
- Nonlinear activation functions such as sigmoids used in classical neural networks are difficult to implement in quantum algorithms since quantum gates only apply linear transformations (a sigmoid is any mathematical function whose graph has a characteristic S-shaped curve).
- Acceleration provided by QML is hard to evaluate, particularly given that all QML techniques are hybrid in nature. In many cases, benchmarks tend to favour a comparison in the quality of the results (e.g. minimising an error function and error rates) rather than proving a quantum speedup.
- QML privacy protection and QML algorithm explainability differs from classical machine learning privacy protection and algorithm explainability, but these topics should nevertheless be adequately addressed.

3. Variational quantum algorithms

Today, many of the quantum algorithms that have been invented are not yet executable on a large problem size on current NISQ quantum computers (Box 3.1). There are simply not enough qubits with high fidelity (error-corrected) qubits available for NISQ quantum computers to be more powerful than classical computers (Figure 3.1).

Noisy Intermediate-Scale Quantum (NISQ) computing is a term, coined by the American theoretical physicist John Phillip Preskill in 2012, that applies to current state-of-the-art quantum computers. The term "noisy" refers to the fact that these quantum computers are very sensitive to perturbances caused by the surrounding environment and may lose their quantum state due to quantum decoherence because they are not sophisticated enough to implement QEC. The term "intermediate-scale" refers to the not-so-large number of qubits.

Box 3.1: Noisy Intermediate-Scale Quantum (NISQ)

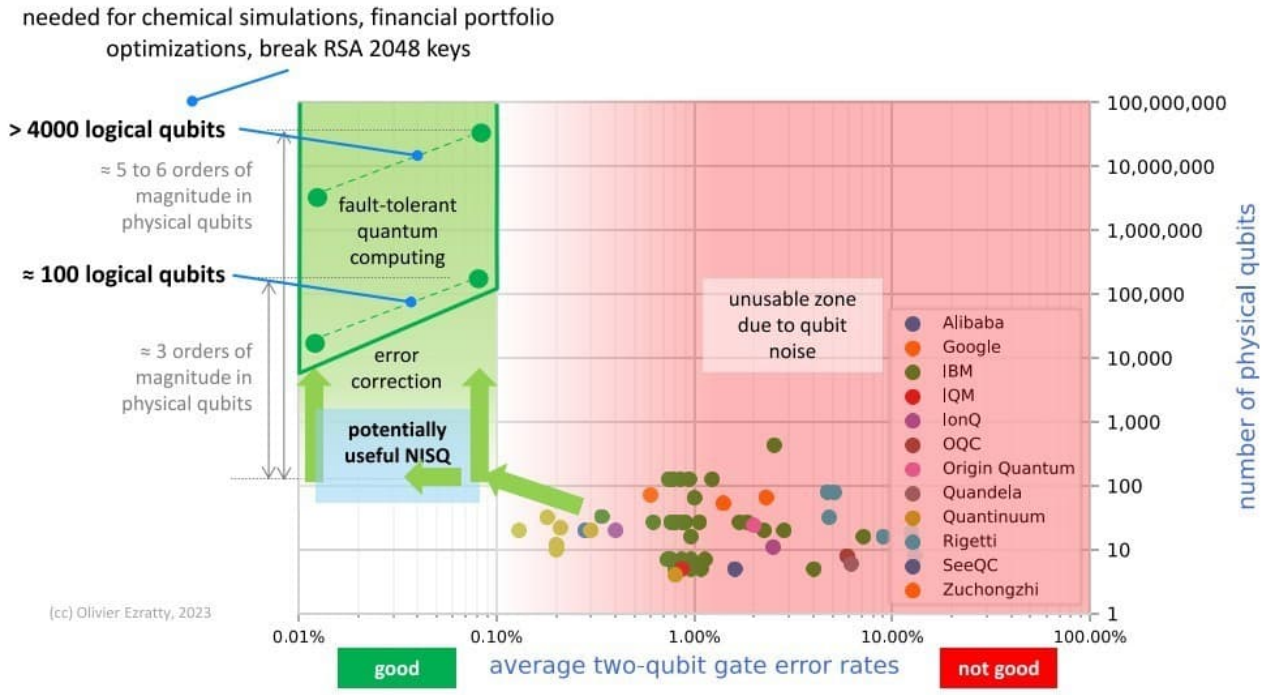

Figure 3.1: FTQC qubit number and fidelity requirements (source: Olivier Ezratty 2023)

Instead of waiting for the availability of powerful FTQC quantum computers, researchers have investigated approaches for taking advantage of currently available NISQ quantum computers.

The most famous of these algorithms are the so-called Variational Quantum Algorithms (VQAs). Many problems of interest, in particular problems in quantum chemistry, can be framed as so-called eigenvalue problems. According to the variational principle of quantum mechanics, the computed energy of the ground (lowest-energy) state of a quantum system decreases as

the approximations to the solution improve, asymptotically approaching the true value from above. This principle has given rise to iterative classical algorithms for solving these problems, where a crude guess of the solution is the input, and a somewhat improved approximation is the output. This output is then used as the guess for the next iteration and with each cycle, the output gets closer and closer to the true solution (but never overshooting).

This approach can be split between a classical and a quantum algorithm, with the iteration step performed by a quantum processor and a classical control step deciding whether to perform another iteration (figure 3.2). The ability to separate the quantum processing among many small, independent steps, with qubit coherence required only over the course of a single quantum computing step, makes this approach a clever way to reduce qubit fidelity requirements and still obtain useful results.

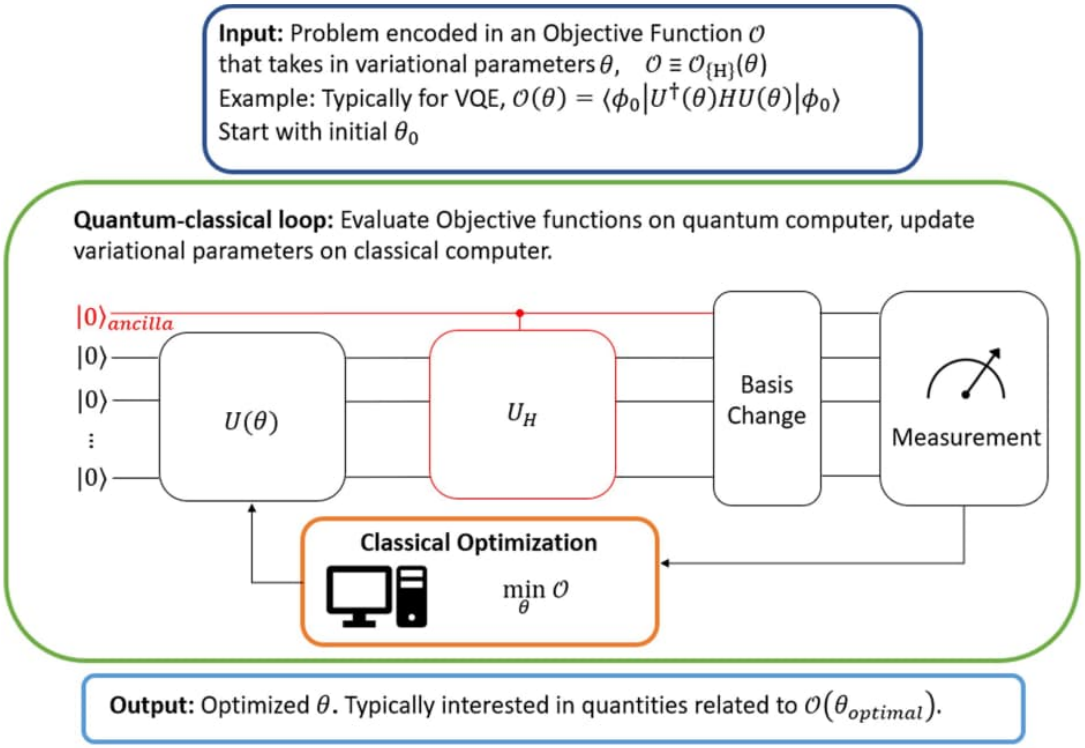

Figure 3.2: VQA components (source: J.W.Z. Lau et al. 2022)

VQAs require a Parameterized Quantum Circuit (PQC) that takes in a set of parameters. It is typically known as the “ansatz” (German for “approach”) as it refers to a trial quantum state used as a starting point for approximations.

The VQA class of quantum algorithms includes:

- **Variational Quantum Eigensolver (VQE)** quantum algorithm for quantum physics simulations, invented in 2013 by Alán Aspuru-Gurzik (Mexican chemical engineer and computer scientist) et al. VQE is a NISQ alternative for QPE which requires a FTQC.

- **Quantum Approximate Optimization Algorithm (QAOA)** quantum algorithm for various combinatorial optimisation tasks, invented in 2014 by the American physicist Edward Henry Farhl. Though NISQ-based QAOA is limited to the use of shallow quantum circuits, it remains a promising candidate for quantum speedup. Its performance can also been improved by using a variant known as adaptive QAOA.

  Note
  A QAOA quantum algorithm often relies on a Quantum Alternating Operator Ansätze (QAOA) component, which is the ansatz quantum circuit that is used by a variational algorithm. Use of the QAOA acronym may therefore be confusing.

  There are some key differences but QAOA also shares similarities with quantum annealing. One looming question is whether QAOA on gate-based universal quantum computers is more efficient than QUBO on quantum annealers.

  Apart from solving combinatorial optimisation problems, QAOA can be generalised to a form that allows for universal quantum computation.

- **Variational Quantum Linear Solver (VQLS)** quantum algorithms for solving linear equations.

- **Variational Quantum Simulator (VQS)** quantum algorithms for simulation of the Hamiltonian evolution of a quantum system.

- **variational QML quantum algorithms** for various machine learning and deep learning tasks.

All of these variational quantum algorithms are heuristic algorithms that provide near-optimal solutions to the problems at hand.

VQA quantum algorithms must overcome the so-called “barren plateau” problem, which prevents convergence unless the ansatz quantum circuit is shallow. It is the equivalent of avoiding local minima traps in classical machine learning when a global minimum is searched but difficult to reach. Barren plateaus are induced by various factors including the number of qubits, gate types, circuit depth, circuit initialisation, measurement types and qubit noise. Research to eliminate this problem is ongoing, e.g. adding additional parameters and constraints to improve gradients in the variational training loop without resorting to inefficient overfitting or with using tensor networks to create the parametrised quantum circuit.

Note
NISQ-based quantum algorithms can be made less susceptible to “noise”, i.e. more resistant to quantum state decoherence, by means of Quantum Error Mitigation (QEM) and/or quantum error suppression (the latter could be implemented in the quantum algorithm itself). QEM denotes a collection of techniques for reducing errors by combining classical post-processing (often based on quantum computation results) with quantum circuit modifications (optimisations). Quantum error suppression denotes a collection of

error reduction that can be implemented in the quantum computer firmware or by appropriate quantum algorithm design.

Another issue with VQA algorithms is that the choice of PQC (ansatz) is often problematic. An optimal PQC should be both expressible, i.e. capable of reaching most parts of the Hilbert space (Box 3.2) and trainable. Unfortunately, the more expressive the PQC is made, the less trainable it becomes.

Hilbert spaces (named after the German mathematician David Hilbert) allow the methods of linear algebra and calculus to be generalised from finite-dimensional Euclidean spaces (named after the ancient Greek mathematician Euclid) to spaces that may be infinite-dimensional. Formally, an Hilbert space is a vector space equipped with an inner product that induces a distance function for which the space is a complete metric space. A qubit state is represented by a vector in a 2-dimensional Hilbert space.

Box 3.2: Hilbert space

Note

Quantum-assisted algorithms have been developed for Hamiltonian simulation on NISQ quantum computers. They are also hybrid quantum-classical algorithms but differ from VQAs in that they (1) do not rely on a classical feedback loop and (2) do not use a PQC (thus avoiding the “barren plateau” problem).

4. Quantum speedup

The quantum speedup, i.e. the acceleration provided by a quantum algorithm compared to the best-in-class classical algorithm for solving a particular "hard" problem, depends on the types of quantum gates used by the quantum algorithm (Figure 4.1).

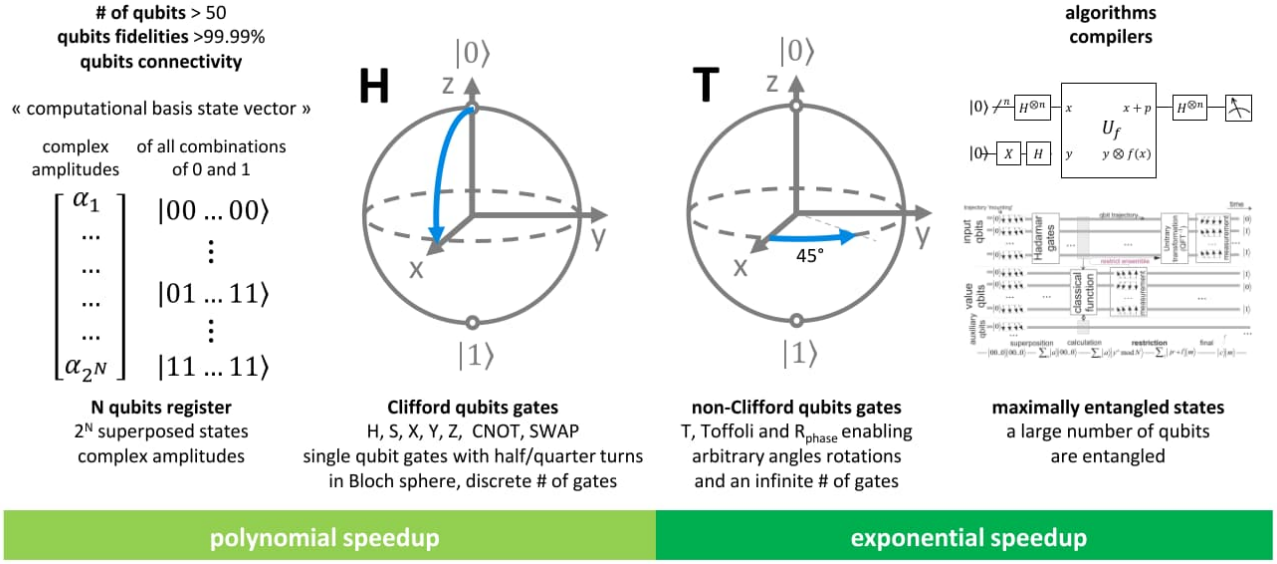

Figure 4.1: Quantum speedup **(source: Olivier Ezratty 2024)**

The Gottesman-Knill theorem (named after the American physicist Daniel Gottesman and the American mathematician and computer scientist Emanuel Knill) states that quantum algorithms using only so-called "digital quantum gates" belonging to the Clifford group (named after the British mathematician and philosopher William Kingdon Clifford) can be emulated in polynomial time on a classical computer. Therefore, non-Clifford quantum gates (so-called "analogue quantum gates") must be used by a quantum algorithm to obtain any quantum speedup.

The Clifford group contains the CNOT gate, the 90º and 180º CR gates, the H gate (named after the French mathematician Jacques Salomon Hadamard), the 90º and 180º R gates, the S gate (aka P gate), the SWAP gate, the T gate, and the X, Y and Z Pauli gates (named after the Austrian physicist Wolfgang Ernst Pauli).

Examples: QFT based algorithms, including Shor's dlog and integer factoring algorithms, provide an exponential speedup because QFT uses R gates other than 90º and 180º, which are non-Clifford quantum gates. Grover's search algorithm provides only a quadratic speedup because it uses H gates, which are Clifford quantum gates.

Note

Even if quadratic speedup is considered as a minor gain in algorithm complexity theory, it can still have a non-negligible practical value and make use of quantum computing worthwhile.

Besides using non-Clifford quantum gates, quantum algorithms also provide exponential speedup if they handle maximally entangled states (Figure 4.1), which means that there is a correlation of states between a set of qubits in the qubit register until the end of the quantum computing. If a quantum algorithm handles islands of disconnected sets of qubits in the qubit register, the speedup will be constrained by the size of these islands. The bigger the island, the bigger the Hilbert space managed by the algorithm. Absolutely Maximally Entangled (AME) states are multipartite quantum states and carry absolute maximum entanglement for all possible qubit register partitions.

Note

While we cannot use entanglement and the quantum teleportation mechanism to transmit classical data faster than light, it appears that entanglement is interconnecting qubits nearly instantaneously during computing while they are affected by qubit gates. This "entanglement nonlocality" is also claimed to provide quantum speedup, but this is still widely disputed among quantum computing experts.

The complexity class of an algorithm as well as the speedup provided by a quantum algorithm is commonly expressed in big *O* notation (Box 4.1), which expresses an upper bound on the time or space complexity of an algorithm in the worst-case scenario.

Big O notation is commonly used to classify algorithms into complexity classes according to how their runtime (time complexity) or memory requirements (space complexity) grow as the problem size (n) grows. It was invented by the German mathematicians Paul Gustav Heinrich Bachman, Edward Landau and a few others.

Box 4.1: Big O notation

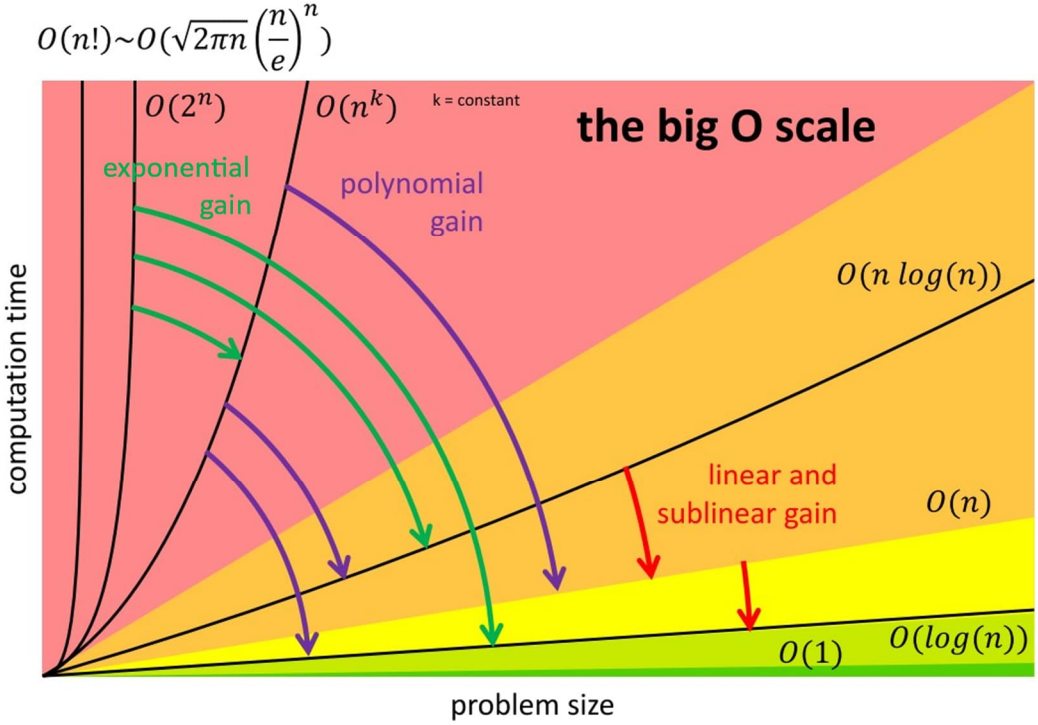

Figure 4.2: Big O algorithm complexity scale (source: Olivier Ezratty 2024)

Note

Except for small problem sizes n, problems of complexity class $O(n!)$ are not solvable.

To outperform their classical counterpart(s), quantum algorithms must demonstrate polynomial speedup or superpolynomial speedup, either weakly superpolynomial or strongly superpolynomial (aka exponential). There only a few dozen known quantum algorithms supposed to be capable of achieving such speedups and furthermore, only a small fraction of these algorithms is commonly used in quantum applications (Figure 4.3).

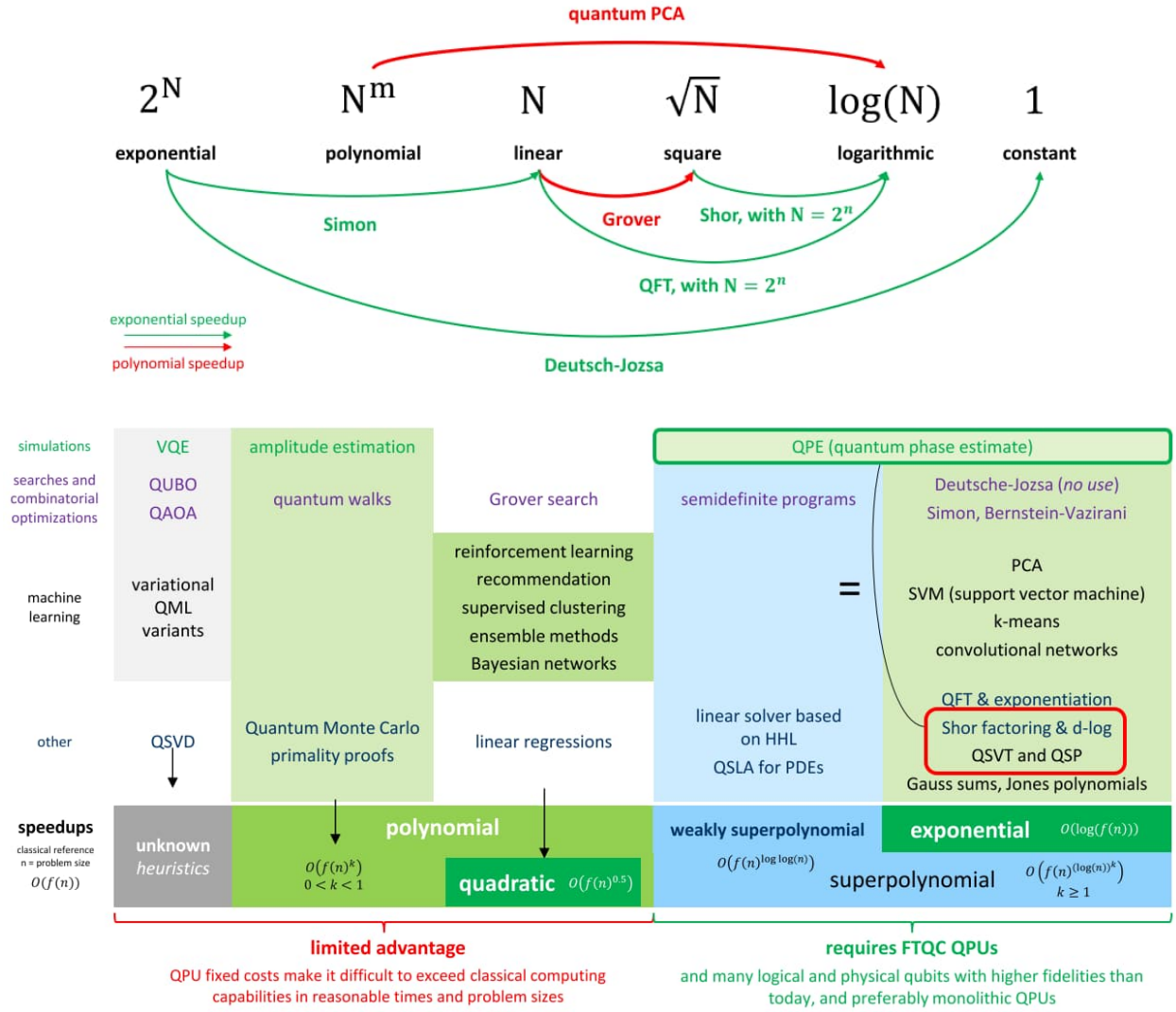

Figure 4.3: Speedups of common quantum algorithms (source: Olivier Ezratty 2024)

Several factors may limit the theoretical speedup provided by a quantum algorithm (Figure 4.4).

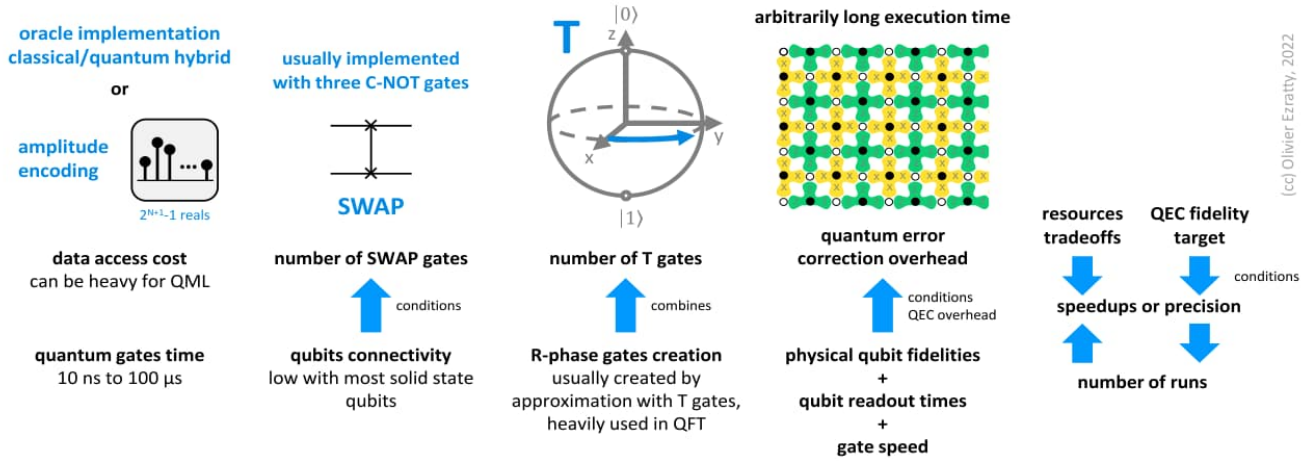

Figure 4.4: Sources of quantum algorithm runtime slowdown (source: Olivier Ezratty 2022)

The following must be taken into account when evaluating quantum speedups claimed by quantum computing technologists:

- One should carefully assess when the quantum speedup will occur with respect to the problem size and corresponding timing. If the quantum algorithm is faster than its best classical counterpart only with timings exceeding human lifetime, use of quantum computing is probably meaningless.

- The quantum versus classical comparison must be based on real-life scenarios and not on worst-case scenarios which can be both unrealistic and unfavourable to classical solutions, thereby overselling the quantum speedup (there can be huge differences between theoretical and practical speedups).

- Quantum speedup will be affected by the number of times the quantum circuit must be run (shot count), which depends on the problem at hand, the number of qubits used, the algorithm output (integers, real numbers or vector state) and whether NISQ or FTQC quantum computers are being used. Assessing the real speedup of a quantum algorithm requires adopting an end-to-end approach considering all the parameters of the quantum algorithm execution time, otherwise projected quantum speedups may not be realistic.

- Overhead imposed by QEC probably increases more than linearly with the number of logical qubits deployed and could cause significant loss of projected quantum speedup at large problem sizes.

- Data preparation (aka data loading) must also be handled, which is of particular importance for QML and oracle-based quantum algorithms, particularly when they rely on data access using qRAM. A problem is that even though quantum computers can use a small number of qubits to represent an exponentially larger amount of data compared to classical computers, there is currently no method to rapidly convert a large amount of classical data to a quantum state. For quantum algorithms that require large inputs, the amount of time needed to create the input quantum state would typically dominate the computation time and greatly reduce

the quantum speedup. This problem does however not exist if the data can be generated algorithmically.

- For small problem sizes, quantum algorithm computing time grows faster than its classical equivalent due to various factors like number of gates per second and data loading time constraints.

- The complexity of some problems can be addressed on conventional computers with probabilistic or heuristic approaches, allowing a significant reduction in the computing time of exponential problems. When moving from this kind of solution to a quantum algorithm, one probabilistic approach is replaced with another one, since quantum computing is also usually highly probabilistic and prone to many computing errors. This complicates the assessment of potential quantum speedups.

- It is very hard to identify the best-in-class classical solution for a given problem, which makes it difficult to perform a meaningful comparison for assessing a potential quantum speedup. The fundamental question is: how can we know that there is no classical algorithm that would have similar scaling behaviour as the quantum algorithm? Though often ignored, this question is key to quantum algorithm research, where often the difficulty is not so much proving that a quantum computer can solve computationally hard problems faster than the best known classical solution (the so-called “gold standard”), but convincingly proving that the latter cannot do the same. It also turns out to be staggeringly hard to prove that problems are indeed computationally hard, as illustrated by the famous P versus NP problem (Box 4.2).

The P versus NP problem is considered by many to be the most important open problem in computer science. It asks whether every problem whose solution can be quickly verified can also be solved quickly. The informal term “quickly” means the existence of an algorithm solving the task that runs in polynomial time (as opposed to exponential time), such that the time to complete the task varies as a polynomial function on the size of the input to the algorithm which solves the problem instance. The class of questions for which some algorithm can provide an answer in polynomial time is P (Polynomial). For some questions, there is no known way to find an answer quickly, but if one is provided with information showing what the answer is, it is possible to verify the answer quickly. The class of questions for which an answer can be verified in polynomial time is NP (Nondeterministic-Polynomial).

Box 4.2: P versus NP problem

- Last but not least, achieving quantum speedup must be considered as shooting on a moving target. On one hand, the speed of quantum algorithm execution is steadily improving with various techniques (faster quantum gates, parallelising quantum computing onto multiple QPUs, design of faster quantum algorithms, etc.). On the other hand, classical algorithms and classical hardware may also improve, and often do so when quantum speedup challenges classical algorithm designers to improve upon the performance of their algorithms (Figure 4.5). Over the past few decades, conjectured quantum speedups have repeatedly gone away when classical algorithms were found with similar performance characteristics (this

phenomenon is called “dequantization”). For example, in 2022 a classical algorithm was discovered that performs Fourier transforms exponentially faster than the FFT algorithm which was the best-known classical algorithm at the time.

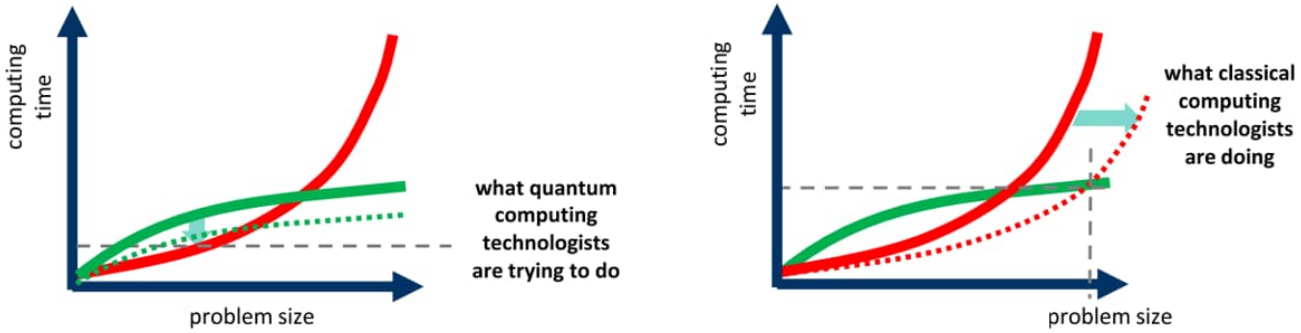

Figure 4.5: Quantum speedup moving target (source: Olivier Ezratty 2024)

Table 4.1 shows the speedup (acceleration) provided by some well-known quantum algorithms.

	algorithm	classical input	quantum input blue for superposed state	quantum output blue for superposed state	classical output	acceleration (# of circuit runs)
FTQC	Deutsche-Jozsa	balanced or unbalanced function in oracle	oracle function	function is balanced if all output qubits are at ground state |0⟩	« yes or no »	exponential (O(1))
	Bernstein-Vazirani	string encoded in a function	can be entirely quantum using a series of quantum gates (with not much real use case) or access some classical data in superposition using a qRAM (which does not exist yet)	(integer) secret string in basis encoding	integer	exponential (O(1))
	Grover	function returning 1 only for one basis		searched item index as integer in basis encoding	integer	quadratic (O(1))
	Simon	periodic function		parameters for a linear equation used to find a period, with average of basis encoding	integer representing function period	exponential (O(1))
	Shor factoring	semi-prime integer	Hadamard gates and parametrized period finding function with exponentiations	regularly spaced amplitudes starting with 0	dividing integer found with continuous fraction post-processing	exponential (depends of period finding integer)
	Shor dlog	two integers				
	QFT	series of values	series of complex amplitudes with amplitude encoding	Fourier coefficients in amplitude encoding, enabling the recovery of the main frequency	main frequency	exponential ($O(1/\epsilon^2)$)
	QPE	Hamiltonian	Hamiltonian encoded in oracle	phase encoded in bitstring	phase as a real number	exponential ($O(1/\epsilon)$)
	HHL	one vector and one matrix	one vector and one matrix amplitude encoding	inverted matrix x entry vector (= one vector) in amplitude encoding	characteristics of the vector to obtain one eigenvalue	exponential (depends)
NISQ	QAOA	objective function to optimize	cost function parameters encoded in an ansatz function (rotation gates and CNOTs)	probabilistic distribution of Pauli strings	cost function value and objective function params	not proven (many)
	VQE	problem Hamiltonian		probabilistic distribution of Pauli strings components of Hamiltonian ground state	cost function evaluation, ansatz update, ground state Hamiltonian	not proven (many due to cost function convergence, Pauli strings # & precision)
	QML classification	depends (training, inference, model)	object vector to classify encoded in amplitude	prediction result as an integer index in basis encoding	integer representing object position in a reference table	depends (many)

Table 4.1: Quantum algorithms and their input and output data (source: Olivier Ezratty 2024)

5. Quantum supremacy and quantum advantage

The term “quantum supremacy” was introduced in 2012 by John Preskill as: “the point where quantum computers can do things that classical computers can’t, regardless of whether those tasks are useful”. According to Preskill’s definition, quantum supremacy refers to a point in time rather than an ongoing phenomenon, but it is of course still a moving target as quantum computing technology is evolving. Google claimed in 2019 to have achieved quantum supremacy but this was only achieved for a very specific esoteric benchmark.

Quantum advantage on the other hand is the goal of demonstrating that a quantum computer can solve a practical problem that no classical computer can solve in any feasible amount of time (Figure 5.1). Conceptually, quantum advantage involves both the engineering task of building a powerful quantum computer and the computational complexity-theoretic task of finding a problem that can be solved by that quantum computer and has a more than polynomial speedup over the best known or possible classical algorithm for that task.

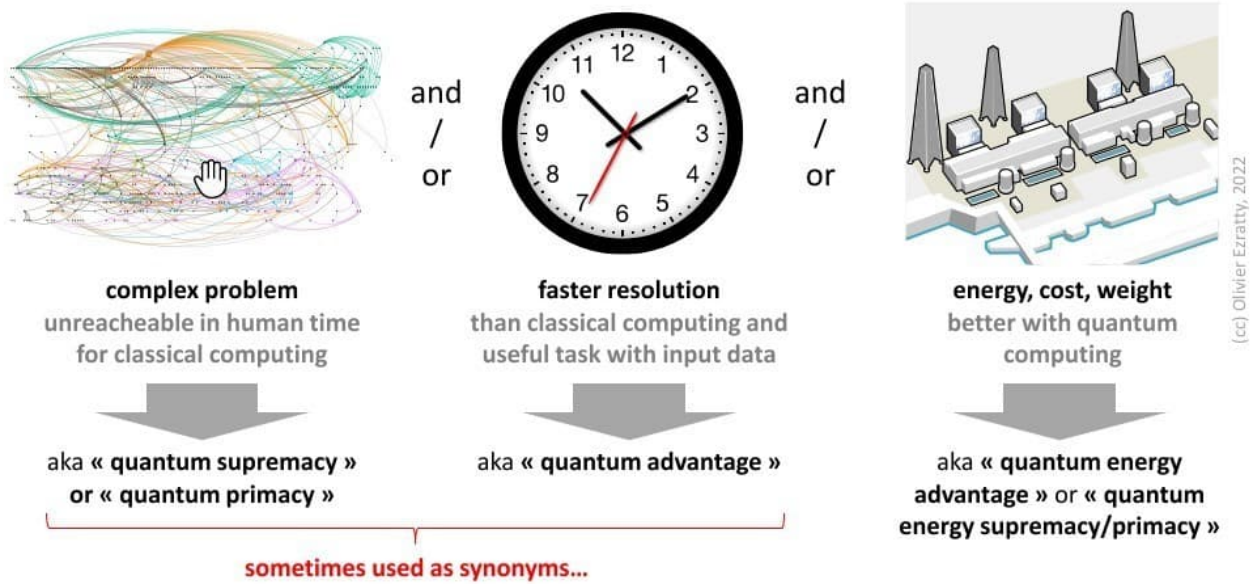

Figure 5.1: Quantum supremacy vs. quantum advantage (source: Olivier Ezratty 2022)

While quantum computers give us the opportunity to directly explore a variety of quantum algorithms and applications, currently available quantum computers have not yet demonstrated quantum advantage with real-world impact, and we are not confident that we have identified an application that will allow us to demonstrate quantum advantage in the short term (Table 5.1).

Quantum supremacy/advantage claims have been proclaimed several times (Table 5.2). In all but a few cases, these claims could not withstand scrutiny by renown quantum computing experts.

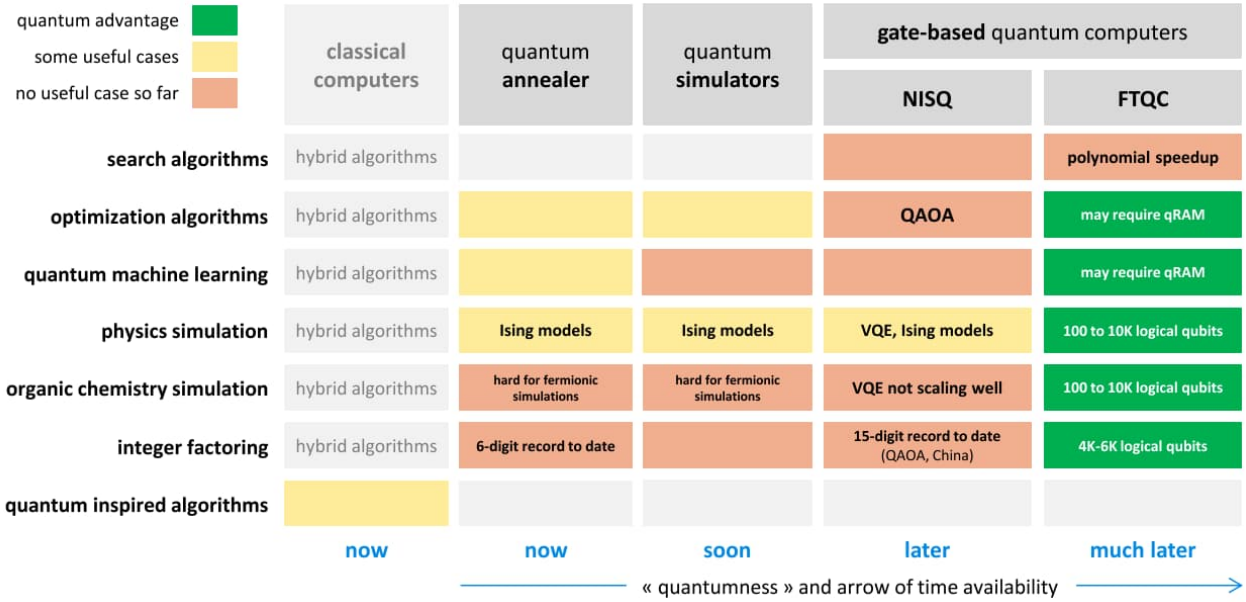

Table 5.1: Quantum computing readiness timeline (source: Olivier Ezratty 2024)

who and when	architecture	algorithm	input data	comment
Google, Oct 2019	Sycamore, 53 superconducting qubits	cross entropy benchmarking	none	running a random gates algorithm
China, December 2020	70 photons modes GBS (Gaussian Boson Sampling)	interferometer photons mixing	none	running a random physical process
IBM Research, December 2020	IBM 27 superconducting qubits	symmetric Boolean functions	SLSB3 function parameters	theoretical demonstration of quantum advantage
Kerenidis, Diamanti et al, March 2021	multi-mode photon dense encoding of verified solution	Quentin Merlin Arthur based verification	output from some quantum computation (not implemented)	no actual computing done in the experiment
China, April 2021	Quantum walk on 62 superconducting qubits	simple quantum walk	simulating a 2-photons Mach-Zehnder interferometer	no quantum advantage at all
University of Arizona, May 2021	supervised learning assisted by an entangled sensor network	variational algorithm, classical computing	data extracted from three entangled squeezed light photonic sensors	not a quantum « computing » advantage per se
China, June 2021	66 superconducting qubits and 110 couplers, Zuchongzhi 1, then 2.1	cross entropy benchmarking	none	56 used qubits
China, September 2021				60 used qubits
China, June 2021	144 photons modes GBS and up to 113 detected events	interferometer photons mixing	none	parametrizable photon phases could lead to a programmable system
Google, AWS, Harvard et al, December 2021	quantum sensors feeding a quantum computer	learning about the principal component of a noisy state	quantum output from quantum sensors	requires some quantum memory
Xanadu June 2022	216 squeezed photons modes GBS (Gaussian Boson Sampling)	time domain multiplexing and interferometer photons mixing	programmable GBS with 1,296 parameters	first programmable GBS
IBM et al, September 2022	hybrid algorithm that could run on NISQ QPUs	QML-TDA unsupervised machine learning technique for extracting valuable shape-related data features	small data sets related to cosmic microwave background	exponential speedup, resilient to noise, requires 96-qubit QPU with 2Q gate and measurement fidelity of 99.99%...
IBM, June 2023	127 qubits	trotterized time evolution of a 2D transverse-field Ising model	none	presented as a utility and not an advantage, quickly classically emulated.
Google, June 2023	70 qubits on 72 Sycamore chipset	cross entropy benchmarking	none	running a random gates algorithm

Table 5.2: Quantum supremacy/advantage claims (source: Olivier Ezratty 2024)

In December 2024, Google claimed that their Willow chip-based quantum computer could perform a Random Circuit Sampling (RCS) benchmark (based on the one used for their 2019 claim) in less than 5 minutes, which would require 10 septillion (10×10^{24}) years on today's fastest supercomputers. Google also suggested that the performance of their quantum computer validates the existence of parallel universes proposed by the American physicist Hugh Everett III and popularised by David Deutsch. Several leading quantum physicists have cautioned that this claim is false.

Despite quantum supremacy/advantage claims, a quantum computer that is powerful and reliable enough to outperform classical computers at practical applications, like simulating chemistry and breaking cryptographic codes, is likely still a long way off.

Some renown scientists, including the Dutch theoretical physicist and Nobel Prize winner Gerardus 't Hooft, the Russian physicist Mikhail Dyakonov and the Israeli mathematician and computer scientist Gil Kalai, caution that building a universal quantum computer is most probably unfeasible because it is not an engineering problem but rather a fundamental scientific problem for which there exists no solution.

Appendix A - References

[Borcherds 2021] The teapot test for quantum computers

[Ezratty 2024] Understanding Quantum Technologies Seventh Edition

[Hidary 2021] Quantum Computing: An Applied Approach Second Edition

[Lau et al. 2022] NISQ computing: where are we and where do we go?

[NOREA 2024] Quantum Annealing Explained

[NOREA 2024] Quantum Computer Benchmarking

[NOREA 2024] Quantum Software Development Tools

[NOREA 2025] Quantum Computing Explained

[Wikipedia 2025]

Appendix B - Acronyms and abbreviations

1Q	one-Qubit quantum gate
2D	two-Dimensional
2Q	two-Qubit quantum gate
µs	microsecond
√	square root
A*	A star
AI	Artificial Intelligence
Aka	also known as
AME	Absolutely Maximally Entangled
ANF	Aramid Nanofiber
AWS	Amazon Web Services
bit	binary digit
BQM	Binary Quadratic Model
C	Celsius
C-NOT	Controlled-NOT gate
CC	Classical-Classical Creative Commons
CNOT	Controlled NOT gate
cons	← *pro et contra*
CQ	Classical-Quantum
CR	Controlled R gate
CSP	Constraint Satisfaction Problem
DFT	Discrete Fourier Transform
DL	Deep Learning
dlog	discrete logarithm
e.g.	exempli gratia
EDP	Electronic Data Processing

et al.	et alia
etc.	et cetera
F	Fahrenheit Fredkin gate
FFT	Fast Fourier Transform
FT	Fourier Transform
FTQC	Fault-Tolerant Quantum Computer
GAN	Generative Adversarial Network
GBS	Gaussian Boson Sampler Gaussian Boson Sampling
GUROBI	G*u*, R*o*thberg and B*i*xby
H	Hadamard gate
HHL	Harrow, Hassidim and Lloyd
i.e.	id est
IBM	International Business Machines
IDA*	Iterative Deepening A star
K	*c*onstant
K	Kilo
Li	Lithium
LLM	Large Language Model
Log	logarithm logarithmic
LP	Linear Programming
MaxCut	Maximum Cut
min	minimum
MIS	Maximal Independent Set

ML	Machine Learning
MIP	Mixed Integer Programming
MZ	Mach-Zehnder
***n*!**	factorial *n*
NISQ	Noisy Intermediate-Scale Quantum
NOREA	Nederlandse Orde van Register EDP-Auditors
np	nanoscale porosity
NP	Nondeterministic-Polynomial
ns	nanosecond
o	objective function
O	big O *o*bjective function
OQC	Oxford Quantum Circuits
OR	Operations Research
P	Phase change gate Polynomial
params	parameters
PCA	Principal Component Analysis
PDE	Partial Differential Equation
PQC	Parameterized Quantum Circuit
pros	← *pro et contra*
Q	Quantum
Q.	Quantum
QAA	Quantum Amplitude Amplification
QAE	Quantum Amplitude Estimation
QAI	Quantum Artificial Intelligence
QAOA	Quantum Alternating Operator Ansätze Quantum Approximate Optimization Algorithm
QC	Quantum-Classical
QCBM	Quantum Circuit Born Machine

QCNN	Quantum Convolutional Neural Network
QCV	Quantum Computer Vision
QEC	Quantum Error Correction
QEM	Quantum Error Mitigation
QFT	Quantum Fourier Transform
QGAN	Quantum Generative Adversarial Network
QGLM	Quantum Generalized Linear Model
QIA	Quantum-Inspired Algorithm
QINN	Quantum Invertible Neural Network
QMARL	Quantum Multi-Agent Reinforcement Learning
QMAS	Quantum Multi-Agent Systems
QMC	Quantum Monte Carlo
QML	Quantum Machine Learning
QML-TDA	QML Topological Data Analysis
QNLP	Quantum Natural Language Processing
qPCA	quantum-inspired Principal Component Analysis
QPE	Quantum Phase Estimation
QPS	Quantum *A*utomated Planning and Scheduling
QPU	Quantum Processing Unit
QQ	Quantum-Quantum
qRAM	quantum Random-Access Memory
QSLA	Quantum Linear Systems Algorithm
QSP	Quantum Signal Processing
QSVD	Quantum Singular Value Decomposer
QSVT	Quantum Singular Value Transformation
qubit	quantum bit
QUBO	Quadratic Unconstrained Binary Optimization
R	Rotational gate
RAM	Random-Access Memory
RCS	Random Circuit Sampling
RNN	Recurrent Neural Network
RSA	Rivest-Shamir-Adleman

RSA-2048	2048-bit RSA
S	*P*hase change gate Sulphur
SAT	S*at*isfiability problem
SDP	Semi*d*efinite Programming
SLSB	Selected Least Significant Bit
SQA	Simulated Quantum Annealing
SVM	Support-Vector Machine
T	*h*alf phase change gate
TABU	TABUlar
TDA	Topological Data Analysis
telecoms	telecommunications
TSP	Travelling Salesperson Problem
U	Unitary
VQA	Variational Quantum Algorithm
VQE	Variational Quantum Eigensolver
VQLS	Variational Quantum Linear Solver
VQS	Variational Quantum Simulator
X	*P*auli X gate
XEB	*C*ross-Entropy Benchmarking
Y	*P*auli Y gate
Z	*P*auli Z gate