Computational Complexity: From Randomness to Quantum Computing

Cristian S. Calude
University of Auckland, New Zealand
www.cs.auckland.ac.nz/~cristian

Guest lecturer: Richard Hua

Second Term, July-August, 2019

Algorithmic randomness

From bits to qubits

Quantum randomness

The quantum gate model

Quantum annealing

Quantum supremacy

Supplementary material

Algorithmic randomness

Random numbers are used extensively in various applications in science, economy and security, ranging from cryptography to numerical simulations.

'Good' randomness is the critical resource for all these applications, but high quality randomness is neither easy to produce nor to certify.

Random numbers had been around for more than 4,000 years, but never have they been in such demand as in our time. The oldest known dice were discovered in a 24th century B.C. tomb in the Middle East.

In 1890 statistician Francis Galton wrote that

> *As an instrument for selecting at random, I have found nothing superior to dice.*

However, the modern world demanded a lot more randomness than dice could offer.

John von Neumann developed a *pseudo-random generator* (PRNG) around 1946: start with an initial random seed value, square it, and slice out the middle digits. A sequence obtained by repeatedly using this method exhibits *some* statistical properties of randomness.

RAND Corporation machine generated numbers using a random pulse generator: its book *A Million Random Digits with 100,000 Normal Deviates* was published in 1955.

In 1951, Ferranti Mark 1 was the first computer with a built-in random number instruction—designed by A. Turing—that could generate 20 random bits at a time using electrical noise.

All modern computers have algorithms that generate *pseudo-random numbers*.

> *I am convinced that the vast majority of my readers, and in fact the vast majority of scientists and even nonscientists, are convinced that they know what 'random' is. A toss of a coin is random; so is a mutation, and so is the emission of an alpha particle.... Simple, isn't it? said Kac in* [34].

Well, no! Kac knew very well that randomness could be called many things, but not simple, and in fact his essay shows that randomness is complicated, and it can be described in more than one way

Books on probability theory do not even attempt to define it. *It's like the concept of a point in geometry books.*

What's wrong with these random numbers?

The pitfalls of low quality randomness have been discovered in the Internet era. An example is the discovery in 2012 of a weakness in the encryption system used worldwide for online shopping, banking and email; the flaw was traced to the numbers a PRNG has produced [37].

Do "better" random numbers exist?
Yes.

Can other methods produce "better" random numbers?
Yes.

Does true or perfect random number exist?
No.

Randomness is best understood through various "symptoms". Here are three of the largely accepted ones:

(i) *Unpredictability:* The impossibility of any agent, acting via uniform, effective means, to predict correctly and reproducibly the outcome of an experiment using finite information extracted from the environment.

(ii) *Incompressibility:* The impossibility to compress a random sequence.

(iii) *Typicality:* Random sequences pass every statistical test of randomness.

The French mathematician Émile Borel, a pioneer of probability theory, argued that there is no way to formalise in an acceptable way the concept of randomness. His argument is based only on one symptom of randomness, typicality, and is known as the *randomness paradox*.

A random bit-string should be "typical": it should not stand out from the crowd of other bit-strings.

Here is Borel's argument. Assume that there is a precise way to distinguish between "random bit-strings" and bit-strings which are "non-random". It does not matter how this criterion was found or operates.

Assume we have a precise **criterion** which can be applied to any bit-string and once a bit-string is given, we can say whether the bit-string is *random* or *non-random*.

Can the **criterion** be consistent? The answer is **negative**.

Indeed, choose the first bit-string which **criterion** asserts it is random. This particular bit-string is

the first bit-string satisfying the property of being random,

a property making it atypical, so non-random!

There are many random number generators:

1. pseudo-random generators, software produced
   - PCG, Random123, xoroshilro128+
2. hardware generators, devices that generate random numbers from physical processes
   - macroscopic, e.g. coin, dice, roulette wheels, lottery machines,
   - microscopic, e.g. thermal noise, photoelectric effect, quantum effects.

In particular, there are many quantum random generators, from lab experiments to openly accessible on the internet and commercial ones (Quantis).

Why do we need another quantum random generator?

Because no current quantum random generator (QRNG) is provably better than the other generators, in particular, pseudo-random generators.

Is this so? Many advantages promised by QRNGs rely on the belief that the outcomes of quantum measurements are

intrinsically/irreducibly unpredictable.

corroborated but not proved by evidence.

This belief underlies:

- the use of QRNGs to produce "quantum random" sequences that are "truly unpredictable",
- the generation of cryptographic keys unpredictable to any adversary.

Is this belief reasonable?

Notation

- Bit strings: ε (empty string), $0, 1, 00, 01, 10, 11, \ldots$.
- $B = \{0, 1\}$ and B^* is the set of all binary strings.
- Strings can be concatenated: from x and y get xy.
- The length of the string x is denoted by $|x|$
  - $|\varepsilon| = 0, |0| = |1| = 1, |00| = |01| = |10| = |11| = 2, \ldots$
  - $|xy| = |x| + |y|$.
- Let $\mathrm{bin}(n)$ be the binary representation of the number $n + 1$ without the leading 1. For example: $1 \mapsto \varepsilon$, $2 \mapsto \ldots$ This is a bijection between non-negative integers and binary strings.
- The quasi-lexicographical order: $x < y$ iff $\mathrm{bin}^{-1}(x) < \mathrm{bin}^{-1}(y)$.

Notation

- A partial function φ from B^* to $B^* \cup \{\infty\}$ is partially computable if there is a Turing machine working to the alphabet B that computes it. By $\mathrm{dom}(\varphi)$ we denote the domain of φ.
- A partially computable function from B^* to B^* is called computable.
- A subset $S \subseteq B^*$ is called computable if its characteristic function is computable.
- A subset $S \subseteq B^*$ is called computably enumerable if there is a computable function f from B^* to B^* such that $f(B^*) = S$, i.e. f enumerates S.

More notation

- B^∞ is the set of all binary infinite sequences $\mathbf{x} = x_1 x_2 \cdots x_n \cdots$.
- $\mathbf{x}(n) = x_1 x_2 \cdots x_n \in B^*$ is the prefix of length $n > 0$ of the sequence $\mathbf{x}$.
- For $w \in B^*$ and $W \subset B^*$, put $wB^\infty = \{\mathbf{x} \in B^\infty \mid w = \mathbf{x}(|w|)\}$ and $WB^\infty = \bigcup_{w \in W} wB^\infty$.
- The family $(wB^\infty, w \in B^*)$ generates a σ-algebra on B^∞.
- The Lebesgue measure is defined by $\mu(wB^\infty) = 2^{-|w|}$.
- A property P of sequences $\mathbf{x} \in B^\infty$ is *true almost everywhere in the sense of* μ in case the set of sequences not having the property P is a null set, i.e. $\mu(\{\mathbf{x} \mid P(\mathbf{x}) \text{ is false}\}) = 0$.

Borel's Law of Large Numbers—the proportion of heads in a "large" number of a fair coin flips "should be" roughly $1/2$—can be expressed by a property true almost everywhere in the sense of μ:

$$\mu\left(\left\{\mathbf{x} \in B^{\infty} \mid \lim_{n\to\infty} \frac{x_1 + x_2 + \ldots + x_n}{n} \neq \frac{1}{2}\right\}\right) = 0.$$

A sequence satisfying a property false almost everywhere with respect to μ is very "particular", "not typical".

Many results in probability theory are true almost everywhere, so it is tempting to say that

> *a sequence* $\mathbf{x}$ *is "random" if it satisfies every property true almost everywhere with respect to* μ*.*

For every sequence $\mathbf{x}$ we can define the property $P_{\mathbf{x}}$ as following:

> $\mathbf{y}$ *satisfies* $P_{\mathbf{x}}$ *if for every* $n \geq 1$ *there exists a natural* $m \geq n$ *such that* $x_m \neq y_m$.

Every $P_{\mathbf{x}}$ is an asymptotic property which is true almost everywhere with respect to μ and $\mathbf{x}$ does not have property $P_{\mathbf{x}}$. Accordingly,

> *no sequence can verify all properties true almost everywhere with respect to* μ.

The above definition is vacuous, hence *there is no true/perfect lawless sequence*.

A "universal" non-trivial property shared by all sequences was discovered by van der Waerden (see for example [30]; for the finite version see ▸ VDWfinite):

> *In every binary sequence at least one of the two symbols must occur in arithmetical progressions of every length.*

There is no constructive proof for van der Waerden result: no algorithm can decide which alternative is true: 0 occurs in arithmetical progressions of every length or 1 occurs in arithmetical progressions of every length.

There is a way to overcome the above difficulty in defining randomness:

> *Consider* only *a sequence of asymptotic properties true almost everywhere.*

But, then which sequence should be chosen?

Clearly, the "larger" the chosen sequence of properties is, the "more random" will be the sequences satisfying those properties.

We also would like to define a notion of randomness such that at least the following properties of unpredictability, incompressibility and typicality are satisfied.

A set S of strings is prefix-free if no string in S is a proper prefix of a string in S.

- The sets $\{x \in B^* \mid |x| = n\}$, $\{1^n 0 \mid n \geq 1\}$ are prefix-free.
- The set $\{x_1 x_1 x_2 x_2 \dots x_n x_n 01 \mid x_1 x_2 \dots x_n \in L\}$ is prefix-free for every $L \subseteq B^*$.
- The set $\{0x_1 0x_2 \dots 0x_n 1 \mid x_1 x_2 \dots x_n \in L\}$ is prefix-free for every $L \subseteq B^*$.
- B^* is not prefix-free.

A prefix-free (or self-delimiting) TM is a TM whose domain is prefix-free.

- The identity $\psi(x) = x$, $x \in B^*$ is not computable by a prefix-free TM; no total function is computable by a prefix-free TM.
- The restriction of the identity $\psi(x) = x$ to the set $\{1^n0 \mid n \geq 1\}$ is computable by a prefix-free TM.
- The function $\theta(1^{|x|}0x) = x$, for all $x \in B^*$ is computable by a prefix-free TM.
- The set of prefix-free TMs is computably enumerable.

UPFTheorem

We can effectively construct a prefix-free TM U such that for every prefix-free TM C there effectively exists a constant $c = c_{U,C}$ such that for each input string x there exists an input z with $|z| \leq |x| + c$ such that $U(z) = C(x)$.

▸ Proof

Let M be a prefix-free Turing machine. The program-size complexity $H_M(x)$ is the size in bits of the smallest program for M to compute x:

$$H_M(x) = \inf\{|p| \mid p \in B^*, M(p) = x\}.$$

When $M = U$ is universal, then H_U is simply denoted by H.

The program-size of the sequence $\mathbf{x}$ is given by the sequence

$$(H(\mathbf{x}(n))_{n>0}.$$

Examples

- *low complexity strings:*

$$H(0^n) \approx \log n + c, H(1^n) \approx \log n + c$$

- *intermediate complexity strings (borderline algorithmically random strings):* if $x^* = \min\{p \mid U(p) = x\}$, then

$$H(x) = |x^*|, H(x^*) \geq |x^*| - c$$

- *high complexity strings (algorithmically random strings):*

$$\max\{H(x) \mid |x| = n\} = n + H(\mathrm{bin}(n)) + c \approx n + 2\log n + c$$

▸ Proof

The binary representation of the number 0.40626 is

$$0.011010000\cdots$$

The complexity of the sequence

$$\mathbf{x} = 011010000\cdots$$

is bounded by a constant, i.e. there exists $c > 0$ such that for each n,

$$H(\mathbf{x}(n)) \leq 2\log n + c.$$

Consider the number π written in binary and let

$$0.\pi_1\pi_2\cdots\pi_n\cdots$$

be the infinite binary sequence of the fractional part of π. There is a constant $c' > 0$ such that for each n,

$$H(\mathbf{\Pi}(n)) \leq 2\log n + c',$$

where

$$\mathbf{\Pi} = \pi_1\pi_2\cdots\pi_n\cdots$$

The *halting problem* requires to construct an algorithm which decides whether an arbitrary Turing machine eventually stops or not.

Theorem [Undecidability of the halting problem]

The halting problem is undecidable.

Proof. Assume by absurdity the existence of a **halting program** deciding the halting problem.

Construct the Turing machine given by the program Trouble(N):

1. `read a natural` N;
2. `generate all programs up to` N `bits in size;`
3. `use the` **halting program** `to check for each generated program whether it halts;`
4. `simulate the running of the above generated programs, and`
5. `compute the biggest value output by these programs, say` o, `and output` $2o + 1$.

The program Trouble(N) **halts** for every natural N.

How long is Trouble(N)? It is about $\log_2 N$ bits.

Reason: to know N we need $\log_2 N$ bits (in binary); the rest is a constant, so Trouble(N) is $\log N + \mathrm{O}(1)$ bits.

There is a big difference between the size—in bits—of $\mathrm{Trouble}(N)$ *and the*
size of the output produced by Trouble(N).

Indeed, for large enough N, Trouble(N) *belongs* to the set of programs having less than N bits (because $\log N + \mathrm{O}(1) < N$).

Accordingly, Trouble(N) *generates itself* at some stage of the computation.

We have got a contradiction since, on one hand, Trouble(N) outputs a natural number no larger than o, but, on the other hand, it outputs $2o + 1$!

As the halting problem is undecidable, we can approach the problem probabilistically. Chaitin's Omega is the "halting probability" of U, that is:

$$\Omega_U = \sum_{p \in \mathrm{dom}(U)} 2^{-|p|}.$$

The Kraft inequality states that for a prefix-free set S,

$$\sum_{p \in S} 2^{-|p|} \leq 1,$$

which implies that $\Omega_U \leq 1$; in fact, $\Omega_U \in (0, 1)$.

Theorem [Omega solves the halting problem]

With the first n bits of Omega we solve the halting problem for U for every program of length less than or equal to n.

▸ Proof

Corollary [Incomputability of Omega]

The sequence $\omega_1\omega_2\cdots\omega_n\cdots$ is not computable.

Some of these bits have actually been determined. For a natural choice of the Turing machine U, the first 40 bits of $\Omega = \Omega_U$ are

$$0001000000010000101001110111000011111010.$$

If we knew the first 5,000 bits, we would know if the Riemann hypothesis is correct. For the Four colour theorem we need even less bits. ▸ OmegaMedia

Corollary [Transcendence of Omega]

The number Ω is transcendental and $\Omega \in (0, 1)$.

Theorem [Omega has maximum complexity]

If

$$\Omega_U = 0.\omega_1\omega_2\cdots\omega_n\cdots$$

then there exits a constant $c > 0$ such that for all $n \geq 1$,

$$H(\omega_1\omega_2\cdots\omega_n) \geq n - c.$$

Reals whose binary expansions are sequences with maximum complexity are called algorithmically random, shortly random.

▸ Proof

A sequence $\mathbf{x} = x_1 x_2 \ldots$ is *bi-immune* if the set $S_{\mathbf{x}} = \{i \geq 1 \mid x_i = 1\}$ and its complement are both immune.

Fact

A sequence $\mathbf{x}$ is bi-immune iff there is no p.c. function φ with infinite domain such that for every $i \in \mathrm{dom}(\varphi)$ we have $x_i = \varphi(i)$.

▸ Proof

Informally, $\mathbf{x}$ is bi-immune if no algorithm can compute exactly more than finitely many bits of $\mathbf{x}_i$.

Corrolary [Bi-immunity of Omega]

The sequence $\omega_1 \omega_2 \cdots \omega_n \cdots$ is bi-immune.

See more in [14, 19].

Theorem [Omega is c.e.]

The number Omega is c.e., that is, the limit of an increasing computable sequence of rationals in $(0, 1)$.

▸ Proof

So, Omega is a c.e. algorithmically random real. In fact, the converse is also true:

Theorem [Omega = c.e. and random]

The set of Omega numbers coincides with the set of c.e. random numbers.

See more in [14, 26].

Corollary

Every pseudo-random sequence is not algorithmic random.

Is algorithmic randomness a good model of randomness?

Algorithmic randomness

- is provably better than pseudo-randomness,
- satisfies the typicality requirement,
- satisfies the incompressibility requirement,
- it is believed to satisfy the requirement of unpredictability, but not convincing model was yet developed to test it,
- no known device for producing it has been yet designed and certified.

See more in [14, 26].

According to Oxford Dictionary, *spurious* means

> *Not being what it purports to be; false or fake. False, although seeming to be genuine. Based on false ideas or ways of thinking.*

The above (dictionary) definition is semantic, hence it depends on an assumed theory: one correlation can be spurious according to one theory, but meaningful with respect to another one.

Can we give a definition of "spurious" which is independent of *any* theory (using formal deductions, mathematical calculations, computer algorithms, etc.)?

In order to satisfy the above broad constraint we define a spurious correlation in a very restrictive way:

> *a correlation is spurious if it appears in a "randomly" generated database.*

A spurious correlation in the above sense is also "spurious" according to any possible definition because, by construction, its values are chosen at "random", as all data in the database. As a consequence, such a correlation cannot provide reliable information on future developments of any type of behaviour. Of course, there are other reasons making a correlation spurious, even within a "non-random" database.

Van der Waerden theorem

[Equidistant correlations everwhere] For any positive integers k there is a positive integer γ such that every bit string of length more than γ contains an arithmetic progression with k occurrences of the same digit or colour, i.e. a monochromatic arithmetic progression of length k.

How "large" is the set of spurious correlations, in the above sense?

Fix a real number α in the open interval $(0, 1)$. A string x of length n can be compressed by αn bits if its complexity $H(x) \leq n - \alpha n$.

The number of strings x of length n which are compressible by αn bits is smaller than

$$2^{(1-\alpha)n} - 1,$$

hence the probability that a string x of length n has $H(x) < n - \alpha n$ is smaller than

$$2^{-\alpha n} \to 0 \ (n \to \infty).$$

See more in [20].

From bits to qubits

Classical computers use bits which are physically represented by two-state classical systems (two positions of an electrical switch, two distinct voltage or current levels allowed by a circuit).

Quantum computers operate with qubits (quantum bits) which are described as quantum states (i.e. abstract, in Hilbert space) that are physically implemented by quantum systems (e.g. an atom) which are in one of two definite states.

For example, the state of a spin-$\frac{1}{2}$ particle, when measured, is always found to be in one of two possible states, represented – using Dirac bra-ket notation – as

$$|+\frac{1}{2}\rangle \text{ (spin-up) or } |-\frac{1}{2}\rangle \text{ (spin-down).}$$

Formally, a qubit is denoted as

$$|0\rangle \text{ or } |1\rangle.$$

Unlike the intermediate states of a classical bit (e.g. any voltages between the "standard" representations of 0 and 1) which can be distinguished from 0 and 1, but do not exist from an informational point of view, quantum intermediate states cannot be reliably distinguished, even in principle, from the basis states, but do have an informational "existence".

A superposition state $|\varphi\rangle$ (pronounced "ket phi") is a qubit state vector represented by a linear combination of basis states conventionally denoted by $|0\rangle$ and $|1\rangle$, that is

$$|\varphi\rangle = \alpha|0\rangle + \beta|1\rangle,$$

where α, β are complex numbers with $\alpha^2 + \beta^2 = 1$.

A qubit is a unit vector in the space $\mathbf{C}^2$, so for each qubit $|x\rangle$, there are two (complex) numbers $a, b \in \mathbf{C}$ such that

$$|x\rangle = a|0\rangle + b|1\rangle = \begin{pmatrix} a \\ b \end{pmatrix}, \tag{1}$$

where

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \ |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix},$$

and $|a|^2 + |b|^2 = 1$.

We can perform a measurement that projects the qubit onto the basis $\{|0\rangle, |1\rangle\}$. Then we will obtain the outcome $|1\rangle$ with probability $|b|^2$, and the outcome $|0\rangle$ with probability $|a|^2$.

With the exception of limit cases $a = 0$ and $b = 0$, the measurement irrevocably disturbs the state:

> *If the value of the qubit is initially unknown, then there is no way to determine a and b with any conceivable measurement.*

However, *after* performing the measurement, the qubit has been prepared in a known state (either $|0\rangle$ or $|1\rangle$); this state is typically different from the previous state.

The above facts point out an important difference between qubits and classical bits. There is no problem in measuring a classical bit without disturbing it, so we can decode all of the information that it encodes. If we have a classical bit with a fixed, but unknown value (0 or 1), then we can only say that there is a probability that the bit has the value 0, and a probability that the bit has the value 1, and these two probabilities add up to 1. When we measure the bit, we acquire additional information; after measurement, we will know completely the value of the bit.

The ability of quantum systems to exist in a "blend" of all their allowed states simultaneously is known as the Principle of Superposition.

Even though a qubit can be put in a superposition (1), it contains no more information than a classical bit, in spite of its having infinitely many states. The reason is that information can be extracted only by measurement. But, as we have argued, for any measurement of a qubit with respect to a given orthonormal basis, there are only two possible results, corresponding to the two vectors of the basis.

Non cloning

It is not possible to capture more information measuring in two different bases because the measurement changes the state. Even worse, quantum states cannot be cloned, hence it's impossible to measure a qubit in two different ways (even, indirectly, by using a copy trick, that is copying and measuring the copy).

Quantum randomness

Questions

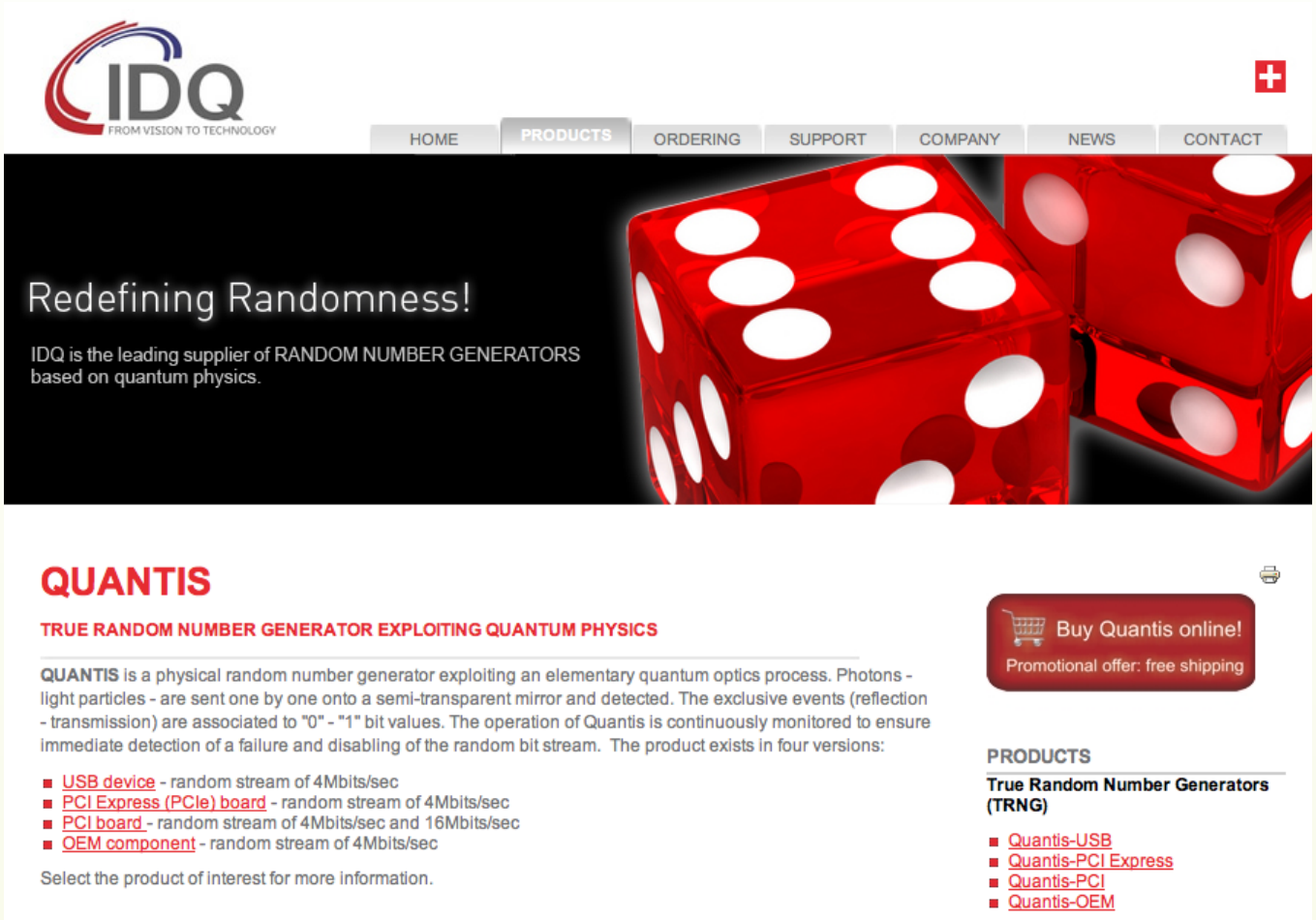
With the understanding acquired on randomness we will look at the best candidate of randomness, quantum randomness.

- How can we produce quantum randomness?
- What is the physical "reason" of quantum randomness?
- What is the quality of quantum randomness?
- Is quantum randomness better than pseudo-randomness?

Quantum randomness is not true randomness

Nature, doi:10.1038/news.2010.181, 14 April 2010.

Quantum randomness is not true randomness



Quantum randomness is not true randomness

More recently, an article published in July 2019 in Wired magazine claims that quantum computers could produce true randomness:

ANIL ANANTHASWAMY SCIENCE 06.23.19 08:00 AM

QUANTUM COMPUTERS COULD BE TRUE RANDOMNESS GENERATORS

The goal is an unending stream of perfectly random 1s and 0s. DAVE WHYTE/QUANTA MAGAZINE

SAY THE WORDS "quantum supremacy" at a gathering of computer scientists, and eyes will likely roll. The phrase refers to the idea that quantum computers will soon cross a threshold where they'll perform with relative ease tasks that are extremely hard for classical computers. Until recently, these tasks were thought to have little real-world use, hence the eye rolls.

Indeterminism and quantum randomness

- Quantum randomness is
  - postulated and
  - generally reduced to the indeterminism of quantum measurements: because the outcome is indeterministic there is no way to predict it, hence it is random

- However, indeterminism does not imply randomness and randomness does not imply indeterminism:
  - pseudo-randomness
  - coin-tossing (chaoticity)
  - Omega number
  - Schrödinger equation

Value indefiniteness is a central notion to this discussion. Informally, for a given quantum system in a particular state, we say that an observable is value definite if the measurement of that observable is predetermined to take a (potentially hidden) value.

In defining it we are guided by Einstein, Podolsky and Rosen famous analysis in [27] which can be encapsulated in the following

> *EPR principle:* *If, without in any way disturbing a system, we can predict with certainty the value of a physical quantity, then there exists a definite value prior to observation corresponding to this physical quantity.*

See more in [6, 8].

EPR principle justifies also

Eigenstate principle: If a quantum system is prepared in a state $|\psi\rangle$, then the projection observable $P_\psi = |\psi\rangle\langle\psi|$ is value definite.

- Admissibility: Definite values must not contradict the statistical quantum predictions for compatible observables on a single quantum.
- Noncontextuality of definite values: The outcome obtained by measuring a value definite observable (a preexisting physical property) is *noncontextual*, i.e. it does not depend on other compatible observables which may be measured alongside the value definite observable.
- Eigenstate principle: If a quantum system is prepared in the state $|\psi\rangle$, then the projection observable $P_\psi = |\psi\rangle \langle\psi|$ is value definite.

Assume the above three hypotheses.

Theorem [6]

Assume a quantum system prepared in the state $|\psi\rangle$ in dimension $n \geq 3$ Hilbert space $\mathfrak{C}^n$, and let $|\phi\rangle$ be any state neither orthogonal nor parallel to $|\psi\rangle$. Then the projection observable P_ψ is value indefinite.

Theorem [8]

The set of value indefinite observables has Lebesgue measure 1, that is, almost all observables are value indefinite.

Value indefiniteness and unpredictability

> *EPR principle:* *If, without in any way disturbing a system, we can predict with certainty the value of a physical quantity, then there exists a definite value prior to observation corresponding to this physical quantity.*

Conversely, if no unique element of physical reality corresponding to a particular physical quantity exists, this is reflected by the physical quantity being value indefinite.

If a physical property is value indefinite we cannot predict with certainty the outcome of any experiment measuring this property.

If we assert of an observable event that it is unpredictable we do not mean, of course, that it is logically or physically impossible for anybody to give a correct description of the event in question before it has occurred; for it is clearly not impossible that somebody may hit upon such a description accidentally. What is asserted is that certain rational methods of prediction break down in certain cases—the methods of prediction which are practised in physical science.

We present a non-probabilistic model of prediction based on the ability of a computable operating agent to correctly predict using finite information extracted from the system of the specified experiment, [9].

Predictions should remain correct in any arbitrarily long (but finite) set of repetitions of the experiment.

Consider a physical experiment E producing a single bit.
We consider an experiment E producing a single bit $x \in \{0, 1\}$.

An example of such an experiment is the measurement of a photon's polarisation after it has passed through a 50-50 beam splitter.

With a particular trial of E we associate the parameter λ (the state of the universe) which fully describes the trial. While λ is not in its entirety an obtainable quantity, we can view it as a resource that one can extract finite information from in order to predict the outcome of the experiment E.

An extractor is a (deterministic) function $\lambda \mapsto \langle\lambda\rangle$ mapping reals to rationals.

For example, $\langle\lambda\rangle$ may be an encoding of the result of the previous trial of E, or the time of day the experiment is performed.

A predictor for E is an algorithm (computable function) P_E which halts on every input and outputs **0** or **1** or **prediction withheld**.

P_E can utilise as input the information $\langle\lambda\rangle$, but, *as required by* EPR, must be passive, i.e. must not disturb or interact with E in any way.

A predictor P_E provides a correct prediction using the extractor $\langle\,\rangle$ for a trial of E with parameter λ if, when taking as input $\langle\lambda\rangle$, it outputs $\mathbf{0}$ or $\mathbf{1}$ and this output is equal to the result of the experiment.

The predictor P_E is $k, \langle\,\rangle$-correct if there exists an $n \geq k$ such that when E is repeated n times with associated parameters $\lambda_1, \ldots, \lambda_n$ producing the outputs $x_1, x_2, \ldots, x_n$, P_E outputs the sequence $P_E(\langle\lambda_1\rangle), P_E(\langle\lambda_2\rangle), \ldots, P_E(\langle\lambda_n\rangle)$ with the following two properties:

(i) no prediction in the sequence is incorrect, and

(ii) in the sequence there are k correct predictions.

The outcome x of a single trial of the experiment E performed with parameter λ is predictable (with certainty) if there exist an extractor $\langle\,\rangle$ and a predictor P_E which is

1. $k, \langle\,\rangle$-correct for all k, and
2. $P_E(\langle\lambda\rangle) = x$.

Theorem 3

If E is an experiment measuring a quantum value indefinite observable, then every predictor P_E using any extractor $\langle\,\rangle$ is not $k, \langle\,\rangle$-correct, for all k.

Theorem 4

In an infinite repetition of E as considered above, no single bit x_i of the generating infinite sequence $x_1 x_2 \ldots$ can be predicted.

epr principle: *If a repetition of measurements of an observable generates a computable sequence, then this implies these observables were value definite.*

Theorem [6]

Assume the epr and Eigenstate principles. An infinite repetition of the experiment E measuring a quantum value indefinite observable generates an incomputable (even bi-immune) infinite sequence $x_1 x_2 \ldots$.

Generalised beam QRNG producing maximum unpredictable, incomputable sequences

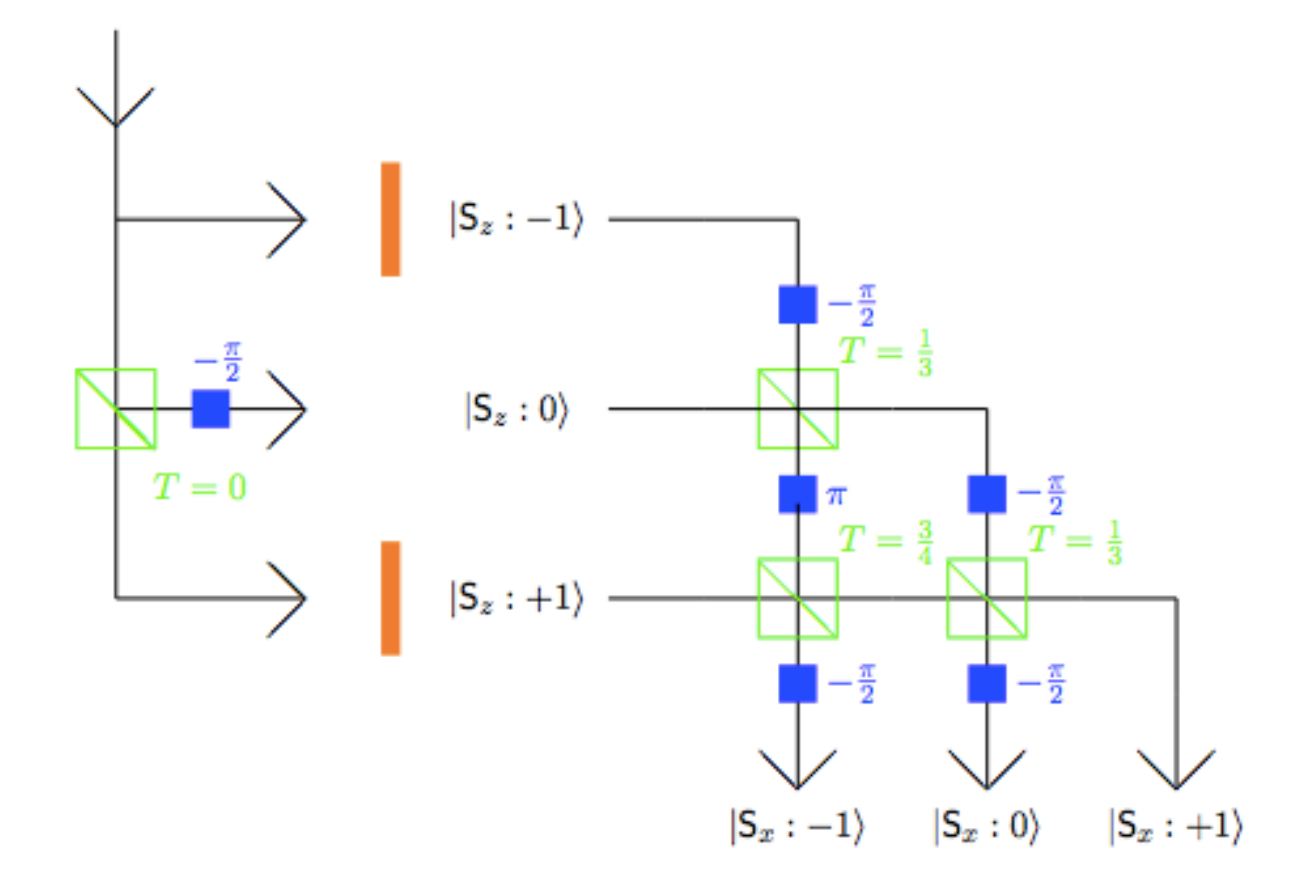


Figure: Generalised beam QRNG

Realisation of the generalised beam QRNG with qutrits

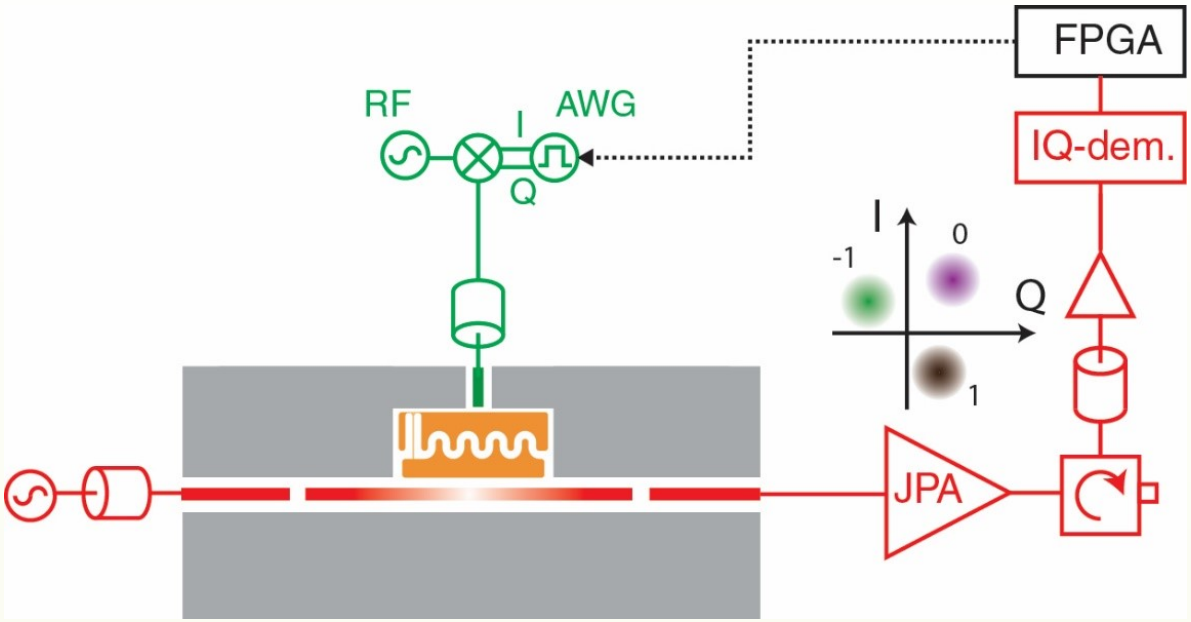


Figure: Realisation of a QRNG with superconducting circuits on a chip (gray). A qutrit (yellow) is coupled to a read out (red) and control circuitries. Near quantum limited Josephson parametric amplifier (JPA) can be used to discriminate all possible outcomes of the protocol with certainty including false runs. Fast programmable gate array (FPGA) can be used to provide the correction pulses to reinitialise the qutrit in its ground state right after the end of the protocol run. [36]

Experimental testing of the quality of the generalised beam QRNG with qutrits

The theory developed above guarantees the bi-immunity and unpredictability of evert sequence produced by he generalised beam QRNG with qutrits.

Can we "probe" these properties on finite prefixes of such sequences? See [7].

The quantum gate model

Quantum computing was first introduced by Yuri I. Manin in 1980 and Richard Feynman in 1982 and intensively researched afterwards.

Quantum algorithms are probabilistic and give the correct answer with high probability; the probability of failure can be decreased by repeating the algorithm.

- In 1985 Deutsch constructed the first model of quantum computer by quantisation of the universal Turing machine.
- In 1994 Shor designed a quantum algorithm for factoring integers in polynomial (quantum) time in the size of the input; the problem whether there is a classical polynomial algorithm for factoring is still open.
- Two years later Grover discovered a quantum algorithm for searching an unsorted N-entry database in $O(\sqrt{N})$ time and $O(\log N)$ space: this algorithm is optimal within the quantum computing model for black box oracles.

The most popular model of quantum computing is probably the *circuit (gate) model* in which quantum algorithms are built from a small set of quantum gates.

Adiabatic quantum computing, proposed in 2000, relies on the adiabatic theorem to do calculations.

A quantum computer is a computation system that makes direct use of quantum-mechanical phenomena, such as superposition and entanglement, to perform operations on data.

Classical computers encode data into binary digits (bits) using classical physical systems – e.g. the position of gear teeth in Babbage's differential engine, a memory element or wire carrying a binary signal, in contemporary machines – which are always in one of two definite states (0 or 1).

Typically, the system is described by one or more continuous parameters, for example, voltage. Such a parameter is used to separate the space into two well-defined regions chosen to represent 0 and 1. Manufacturing imperfections, local perturbations may affect, so signals are periodically restored toward these regions to prevent them from drifting away.

References: [21, 31, 42].

The quantum evolution of a qubit is described by a "unitary operator", that is an operator induced by a unitary matrix.[1]

Any unitary operator $U : \mathbf{C}^2 \to \mathbf{C}^2$ can be viewed as a single qubit gate. Considering the basis $\{|0\rangle, |1\rangle\}$, the transformation is fully specified by its effect on the basis vectors. In order to obtain the associated matrix of an operator U, we put the coordinates of $U|0\rangle$ in the first column and the coordinates of $U|1\rangle$ in the second one.

[1] A quadratic matrix A of order n over $\mathbf{C}$ is *unitary* if $AA^\dagger = I$ (the identity $n \times n$ matrix); $A^\dagger$ is the transposed conjugate matrix of A.

So, the general form of a transformation that acts on a single qubit is a 2×2 matrix

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix},$$

which transforms the qubit state $\alpha|0\rangle + \beta|1\rangle$ into the state $(\alpha a + \beta b)|0\rangle + (c\alpha + d\beta)|1\rangle$:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \alpha a + \beta b \\ c\alpha + d\beta \end{pmatrix}.$$

We may think of logic gates as transformations. For example, the NOT transformation which interchanges the vectors $|0\rangle$ and $|1\rangle$, is the matrix

$$\text{NOT} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

It flips that state of its input,

$$\text{NOT}\,|0\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |1\rangle,$$

and

$$\text{NOT}\,|1\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = |0\rangle.$$

The square-root of NOT (introduced by Deutsch) is the transformation

$$\sqrt{\text{NOT}}: \begin{array}{rcl} |0\rangle & \rightarrow & \frac{1}{2}(1+i)|0\rangle + \frac{1}{2}(1-i)|1\rangle, \\ |1\rangle & \rightarrow & \frac{1}{2}(1-i)|0\rangle + \frac{1}{2}(1+i)|1\rangle, \end{array}$$

$$\sqrt{\text{NOT}} = \frac{1}{2}\begin{pmatrix} 1+i & 1-i \\ 1-i & 1+i \end{pmatrix}.$$

$$\sqrt{\text{NOT}} \cdot \sqrt{\text{NOT}} = \text{NOT}\ , \tag{2}$$

and

$$\sqrt{\text{NOT}} \cdot \sqrt{\text{NOT}}^{\dagger}$$

$$= \frac{1}{4}\begin{pmatrix} 1+i & 1-i \\ 1-i & 1+i \end{pmatrix}\begin{pmatrix} 1-i & 1+i \\ 1+i & 1-i \end{pmatrix} = I.$$

The square-root of NOT is a typical "quantum" gate in the sense that it is impossible to have a single-input/single-output classical binary logic gate that satisfies (2). Indeed, any classical binary

$$\sqrt{\text{NOT}}_{\text{classical}}$$

gate is going to output a 0 or a 1 for each possible input 0/1. Assume that we have such a classical square-root of NOT gate acting as a pair of transformations

$$\sqrt{\text{NOT}}_{\text{classical}}(0) = 1,\ \sqrt{\text{NOT}}_{\text{classical}}(1) = 0.$$

Then, two consecutive applications of it will *not* flip the input!

Finally we consider the Hadamard transformation H is defined by

$$H: \begin{array}{rcl} |0\rangle & \rightarrow & \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \\ |1\rangle & \rightarrow & \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \end{array},$$

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

This transformation has a number of important applications. When applied to $|0\rangle$, H creates a superposition state

$$\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle).$$

The simplest way to illustrate the power of quantum computing is to solve the so-called Deutsch's problem. Consider a Boolean function $f : \{0,1\} \to \{0,1\}$ and suppose that we have a black box to compute it. We would like to know whether f is constant (that is, $f(0) = f(1)$) or balanced ($f(0) \neq f(1)$). To make this test classically, we need two computations of f, $f(0)$ and $f(1)$ and one comparison. Is it possible to do it better? The answer is affirmative, and here is a quantum solution.

Suppose that we have a quantum black box to compute f. Consider the transformation U_f which applies to two qubits, $|x\rangle$ and $|y\rangle$ and produces $|x\rangle|y \oplus f(x)\rangle$.[2] This transformation flips the second qubit if f acting on the first qubit is 1, and does nothing if f acting on the first qubit is 0.

[2] By $\oplus$ we denote, as usual, the sum modulo 2.

The black box is "quantum", so we can chose the input state to be a superposition of $|0\rangle$ and $|1\rangle$. Assume first that the second qubit is initially prepared in the state $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. Then,

$$\begin{aligned} U_f\left(|x\rangle\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\right) &= |x\rangle\frac{1}{\sqrt{2}}(|0 \oplus f(x)\rangle - |1 \oplus f(x)\rangle) \\ &= (-1)^{f(x)}|x\rangle\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle). \end{aligned}$$

Next take the first qubit to be $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. The black box produces

$$
\begin{aligned}
& U_f \left(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \right) \\
&= \frac{1}{\sqrt{2}}((-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle) \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \\
&= \frac{1}{2}(-1)^{f(0)}(|0\rangle + (-1)^{f(0)\oplus f(1)}|1\rangle)(|0\rangle - |1\rangle).
\end{aligned}
$$

Next perform a measurement that projects the first qubit onto the basis

$$\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle).$$

Neglecting the sign and normalisation, we obtain

- $(|0\rangle - |1\rangle)$ if the function f is balanced and
- $(|0\rangle + |1\rangle)$ in the opposite case.

The quantum algorithm solving Deutsch's problem is:

1. Start with a closed physical system prepared in the quantum state $|01\rangle$.
2. Evolve the system according to H.
3. Evolve the system according to U_f.
4. Evolve the system according to H.
5. Measure the system. If the result is the second possible output, the f is constant; if the result is the fourth possible output, then f is balanced.

To prove the correctness of the quantum algorithm, we shall show that the first and third possible outputs can be obtained with probability zero, while one (and only one) of the second and the fourth outcomes will be obtained with probability one, and the result solves correctly Deutsch's problem.

To this aim we follow step-by-step the quantum evolution described by the above algorithm.

In Step 1 we start with a closed physical system prepared in the quantum state $|01\rangle$:

$$V = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}.$$

After Step 2 the system has evolved in the state (which is independent of f):

$$HV = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \end{pmatrix} \times \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{2} \\ -\frac{1}{2} \\ \frac{1}{2} \\ -\frac{1}{2} \end{pmatrix}.$$

After Step 3 the quantum system is in the state (which *depends* upon f):

$$U_f HV = \begin{pmatrix} 1-f(0) & f(0) & 0 & 0 \\ f(0) & 1-f(0) & 0 & 0 \\ 0 & 0 & 1-f(1) & f(1) \\ 0 & 0 & f(1) & 1-f(1) \end{pmatrix} \times \begin{pmatrix} \frac{1}{2} \\ -\frac{1}{2} \\ \frac{1}{2} \\ -\frac{1}{2} \end{pmatrix}$$

$$= \begin{pmatrix} \frac{1}{2}-f(0) \\ -\frac{1}{2}+f(0) \\ \frac{1}{2}-f(1) \\ -\frac{1}{2}+f(1) \end{pmatrix}.$$

After Step 4, the quantum state of the system has become:

$$HU_f HV = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \end{pmatrix} \times \begin{pmatrix} \frac{1}{2} - f(0) \\ -\frac{1}{2} + f(0) \\ \frac{1}{2} - f(1) \\ -\frac{1}{2} + f(1) \end{pmatrix}$$

$$= \begin{pmatrix} 0 \\ 1 - f(0) - f(1) \\ 0 \\ f(1) - f(0) \end{pmatrix}.$$

Finally, in Step 5 we *measure* the current state of the system, that is, the state $HU_f HV$, and we get:

1. output 1 with probability $p_1 = 0$,
2. output 2 with probability $p_2 = (1 - f_Q(|0\rangle) - f_Q(|1\rangle))^2$,
3. output 3 with probability $p_3 = 0$,
4. output 4 with probability $p_4 = (f_Q(|1\rangle) - f_Q(|0\rangle))^2$.

To conclude:

- if $f_Q(|0\rangle) = f_Q(|1\rangle)$, then $f(0) + f(1) = 0 \pmod 2$, $f(1) - f(0) = 0$; consequently, $p_2 = 1, p_4 = 0$.
- if $f_Q(|0\rangle) \neq f_Q(|1\rangle)$, then $f(0) + f(1) = 1$, $f(1) - f(0) = -1$ or $f(1) - f(0) = 1$; consequently, $p_2 = 0, p_4 = 1$.
- the outputs 1 and 3 have each probability zero.

So, Deutsch's problem was solved with only one computation of f. The explanation consists in the ability of a quantum computer to be in a blend of states: we can compute $f(0)$ and $f(1)$, but also, and more importantly, we can extract some information about f which tells us whether $f(0)$ is equal or not to $f(1)$.

Can we match classically this complexity?

The claim made more than 30 years ago in [24] and still maintained in 2019 is **negative**. This claim **false** appears in many books on quantum computing with the notable exception of [42]. See more in [16, 4, 5, 33].

Deutsch's problem was solved with only one use of U_f. The solution is probabilistic, and the result is obtained with probability one. Its success relies on the following three facts:

- the "embedding" of f into f_Q (see also the discussion in Mermin [41], end of section C, p. 11),
- the ability of the quantum computer to be in a superposition of states: we can check whether $f_Q(|0\rangle)$ is equal or not to $f_Q(|1\rangle)$ not by computing f_Q on $|0\rangle)$ and $|1\rangle$, but on a superposition of $|0\rangle)$ and $|1\rangle$, and
- the possibility to extract the required information with just one measurement.

We *de-quantise Deutsch's algorithm*, see [16, 33], in the following way. We consider $\mathbf{Q}$ the set of rationals, and the space $\mathbf{Q}[i] = \{a + bi \mid a, b \in \mathbf{Q}\}, (i = \sqrt{-1})$. We embed the original function f in $\mathbf{Q}[i]$ and we define the classical analogue C_f of the quantum evolution U_f acting from $\mathbf{Q}[i]$ to itself as follows:

$$C_f(a + bi) = (-1)^{0 \oplus f(0)} a + (-1)^{1 \oplus f(1)} bi. \tag{3}$$

The four different possible bit-functions f induce the following four functions C_f from $\mathbf{Q}[i]$ to $\mathbf{Q}[i]$ ($\bar{x}$ is the conjugate of x):

$$\begin{aligned} C_{00}(x) &= \bar{x}, \text{ if } f(0)=0, f(1)=0, \\ C_{01}(x) &= x, \text{ if } f(0)=0, f(1)=1, \\ C_{10}(x) &= -x, \text{ if } f(0)=1, f(1)=0, \\ C_{11}(x) &= -\bar{x}, \text{ if } f(0)=1, f(1)=1. \end{aligned}$$

Deutsch's problem becomes the following:

> *A function f is chosen from the set $\{C_{00}, C_{01}, C_{10}, C_{11}\}$ and the problem is to determine, with a single query, which type of function it is, balanced or constant.*

The following *deterministic* classical algorithm solves the problem:

`Given` f`, calculate` $(i-1)\times f(1+i)$`. If the result is real, t`
`is balanced; otherwise, the function is constant.`

Indeed, the algorithm is correct because if we calculate $(i-1)\times f(1+i)$ we get:

$$(i-1)\times C_{00}(1+i) = (i-1)(1-i) = 2i,$$

$$(i-1)\times C_{01}(1+i) = (i-1)(1+i) = -2,$$

$$(i-1)\times C_{10}(1+i) = (i-1)(-1-i) = 2,$$

$$(i-1)\times C_{11}(1+i) = (i-1)(i-1) = -2i.$$

If the answer is real, then the function is balanced, and if the answer is imaginary, then the function is constant.

Two-dimensionality can be obtained in various other simpler ways.

For example, we can choose as space the set $\mathbf{Z}[\sqrt{2}] = \{a + b\sqrt{2} \mid a, b \in \mathbf{Z}\}$, where $\overline{a + b\sqrt{2}} = a - b\sqrt{2}$.

Using a similar embedding function as (3),

$$C_f(a + b\sqrt{2}) = (-1)^{0 \oplus f(0)} a + (-1)^{1 \oplus f(1)} b\sqrt{2},$$

now acting on $\mathbf{Z}[\sqrt{2}]$, we get the solution:[3]

`Given` f`, calculate` $(\sqrt{2} - 1) \times f(1 + \sqrt{2})$.
`If the result is rational, then the function is balanced; otherwise, the function is constant.`

[3] In fact we don't need the whole set $\mathbf{Z}[\sqrt{2}]$, but its finite subset $\{a + b\sqrt{2} \mid a, b \in \mathbf{Z}, |a|, |b| \leq 3\}$.

Is the quantum gate model Turing complete?

Can the quantum gate model simulate any Turing machine?

No.

The reason is that all function computable by the quantum gate model are **total**. The halting problem for the quantum gate model is trivial. See more in [11].

Quantum annealing

Adiabatic quantum computing (AQC) is an alternative to the quantum gate model. Results by [28, 10] "suggest" that the two models of quantum computing are polynomially equivalent.

More general references about AQC and D-Wave machine: [39, 17, 23]. A presentation of practical QA is in [40].

One distinction between the two models is their discrete versus analog natures: the quantum gate mode is discrete while ADC is continuum.

Adiabatic in AQC refers to a process in which there is no transfer of energy between the system and its environment. This a thermal (not quantum) property; its name is used here metaphorically.

AQC uses the propensity of physical systems – classical or quantum – to minimise their free energy. *Quantum annealing* is free energy minimisation in a quantum system.

An AQC algorithm that computes an exact or approximate solution of an optimisation problem encoded in the ground state – its lowest-energy state – of a Hamiltonian (the operator corresponding to the total energy of the system).

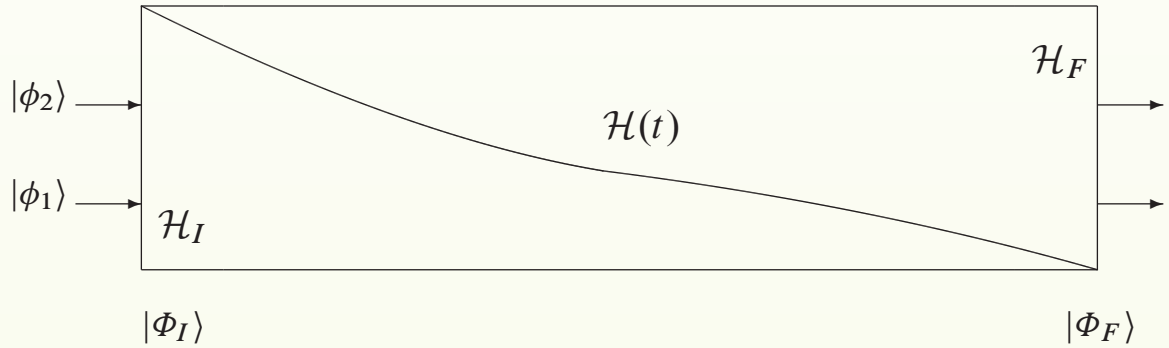


Figure: AQC example

The algorithm starts at an initial state H_I that is easily obtained, then evolves adiabatically by changing to the Hamiltonian H_F. The qubit states are read at the end of the transition.

An example of evolution is $H = (1 - t)H_I + tH_F$ as the time t increases monotonically from 0 to 1. During the entire computation, the system must stay in a valid ground state.

If the system can reach its ground state we get an exact solution; if it can only reach a local minimum, then we get an approximate solution.

The slower the evolution process the better the approximate (possibly exact) solution is obtained.

AQC is based on the 1928 Born-Fock adiabatic theorem which accounts for the adiabatic evolution of quantum states when the change in the time-dependent Hamiltonian is sufficiently slow:

> *A physical system remains in its instantaneous eigenstate if a given perturbation is acting on it slowly enough and if there is a gap between the eigenvalue and the rest of the Hamiltonian's spectrum.*

Quantum Annealing

- A heuristic approach originally designed for solving hard combinatorial optimization problems, it is now understood as a special type of AQC.
- Not as powerful as AQC in general (i.e. QA can not simulate arbitrary quantum gate).
- Debatable whether it can provide any actual speedup over classical algorithms.
- Relatively easy to physically engineer.

The D-Wave computers are produced by the Canadian company D-Wave Systems.

D-Wave One (2011) operates on a 128 qubit chipset.

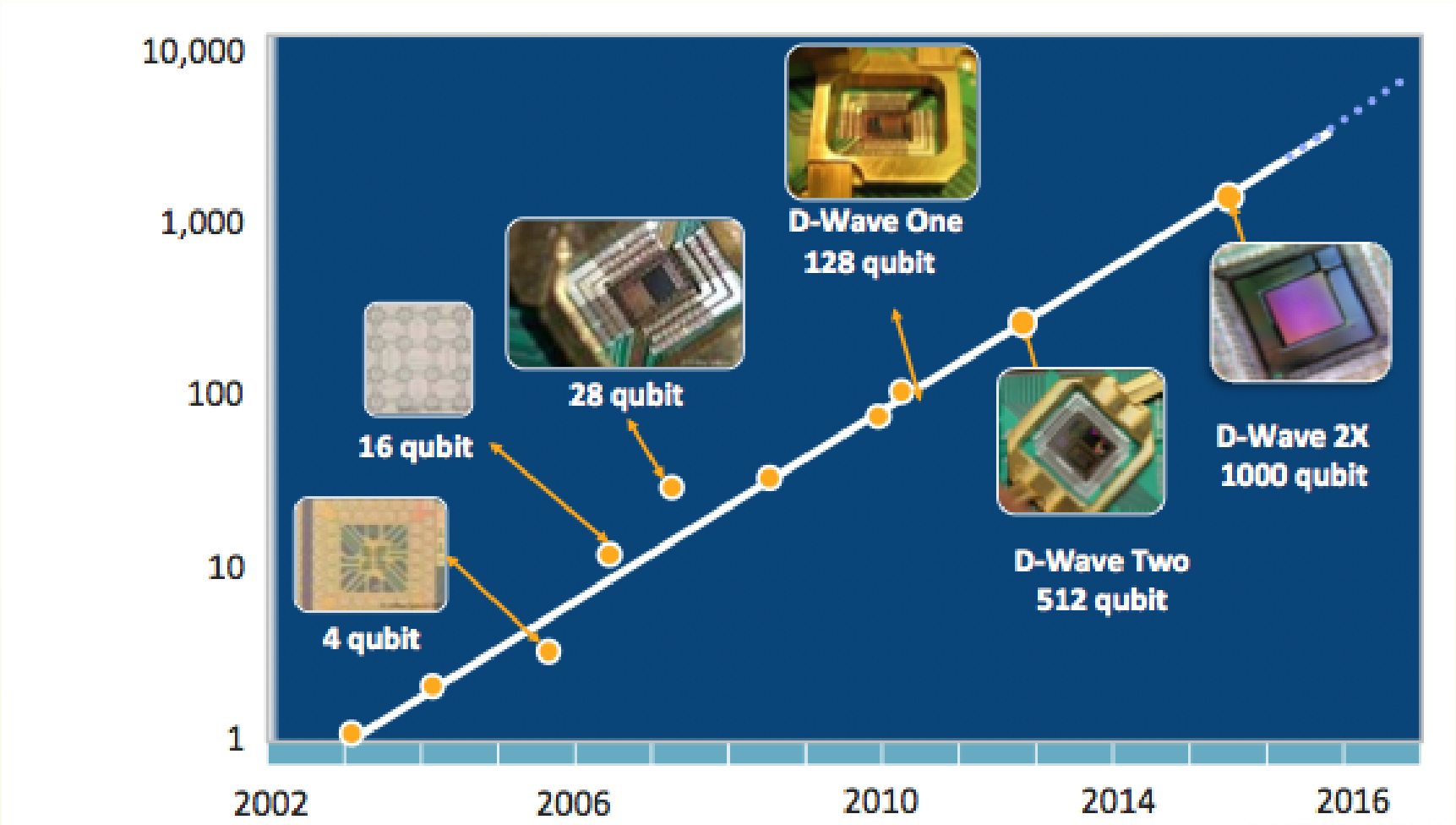
D-Wave Two (2013) works with 512 qubits.

D-Wave 2X (2015) is a 1024 qubit quantum computer.

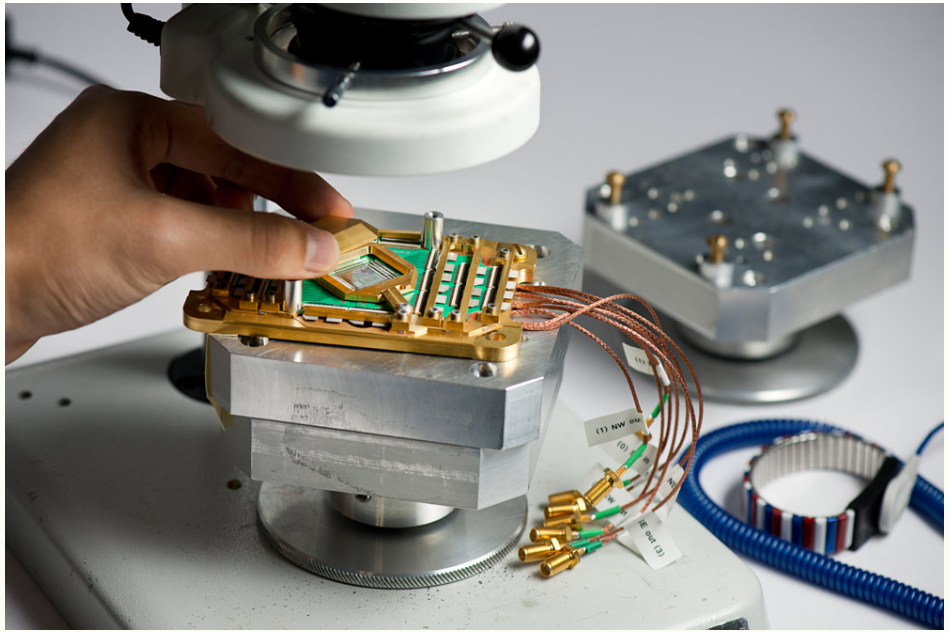
D-Wave 2000Q (2017) operates with 2048 qubits.

D-Wave Systems describes their machines as using quantum annealing to improve convergence of the system's energy towards the ground state energy of an QUBO problem.

Moore-like growth in hardware (number of qubits)



D-Wave 2000Q



The Quadratic Unconstrained Binary Optimisation (QUBO) is an **NP**-hard mathematical problem consisting in the minimisation of a quadratic objective function

$$z = \mathbf{x}^T Q \mathbf{x},$$

where $\mathbf{x}$ is a n-vector of binary variables and Q is a symmetric $n \times n$ matrix:

$$x^* = \min_{\mathbf{x}} \sum_{i \geq j} x_i Q_{(i,j)} x_j, \text{ where } x_i \in \{0, 1\}.$$

D-Wave solves only one problem: QUBO. Really?

The computer architecture consists of qubits arranged with a host configuration as a subgraph of a Chimera graph which consists of an $M \times N$ two-dimensional lattice of blocks, with each block consisting of $2L$ vertices (a complete bipartite graph $K_{L,L}$), in total $2MNL$ variables.

The D-Wave One has $M = N = L = 4$ for a maximum of 128 qubits.

D-Wave qubits are loops of superconducting wire, the coupling between qubits is magnetic wiring and the machine itself is supercooled.

To index a qubit we use four numbers (i, j, u, k), where

- (i, j) indexes the (row, column) of the block,
- $u \in \{0, 1\}$ is the left/right bipartite half of $K_{L,L}$ and
- $0 < k < L$ is the offset within the bipartite half.

Qubits indexed by (i, j, u, k) and (i', j', u', k') are neighbours if and only if

1. $i = i'$ and $j = j'$ and $[(u, u') = (0, 1)$ or $(u, u') = (1, 0)]$ or
2. $i = i' \pm 1$ and $j = j'$ and $u = u'$ and $u = 0$ and $k = k'$ or
3. $i = i'$ and $j = j' \pm 1$ and $u = u'$ and $u = 1$ and $k = k'$.

D-Wave Chimera graph

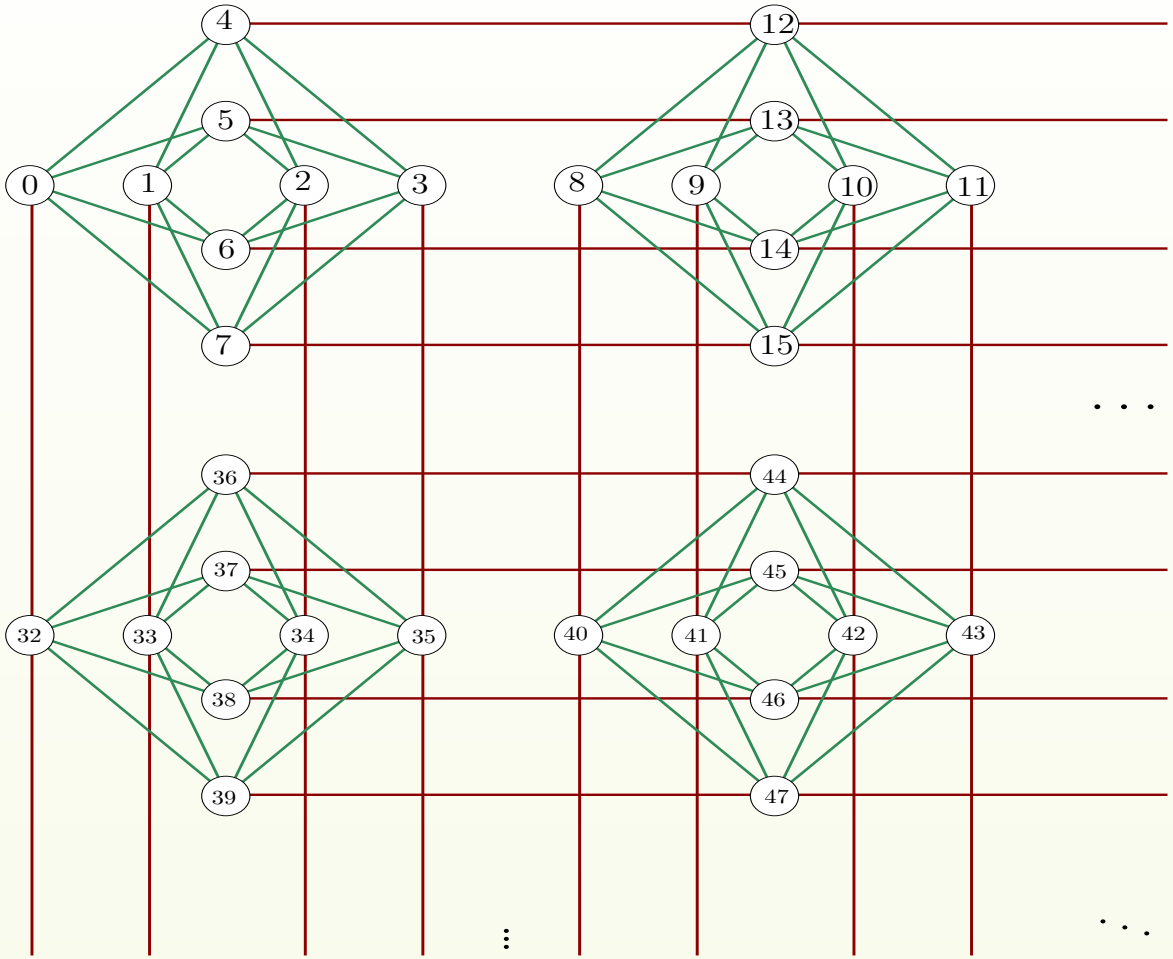


Figure: D-Wave architecture: a subgraph of a Chimera graph with $L = N = 4$.

In the Figure above shows for $L = N = 4$ (and $M > 2$) the structure of an initial part of a Chimera graph where the two half partitions of the bipartite graphs $K_{L,L}$ (blocks) are drawn horizontally and vertically, respectively.

The linear index (qubit id of the vertices) from the four tuple (i, j, u, k) is the value $2NLi + 2Lj + Lu + k$.

Problem solving workflow

1. State the problem. (Example: maximum-independent-set problem.)
2. Reformulate the problem as a QUBO. (Example: reformulate the integer programming optimisation problem as an equivalent QUBO.)
3. Decompose a large problem. (Example: use branch-and-bound methods to divide a large problem into smaller subproblems.)
4. Minor embedding problem. (Example: minor embed the QUBO graph into a Chimera graph.)
5. Submitting the problem to the D-Wave machine.

Maximum Weighted Independent Set (MWIS) Problem:

Input: A graph $G = (V, E)$ with positive vertex weights $w : V \to \mathbb{R}^+$.

Task: Find an independent set $V' \subseteq V$ such that $\sum_{v \in V'} w(v)$ is as large as possible.

Fact

For every graph $G = (V, E)$ with positive vertex weights $w : V \to \mathbb{R}^+$, the largest weighted independent set of G is equal to $\sum_{v \in V} w(v)$ minus the smallest weighted vertex cover of G.

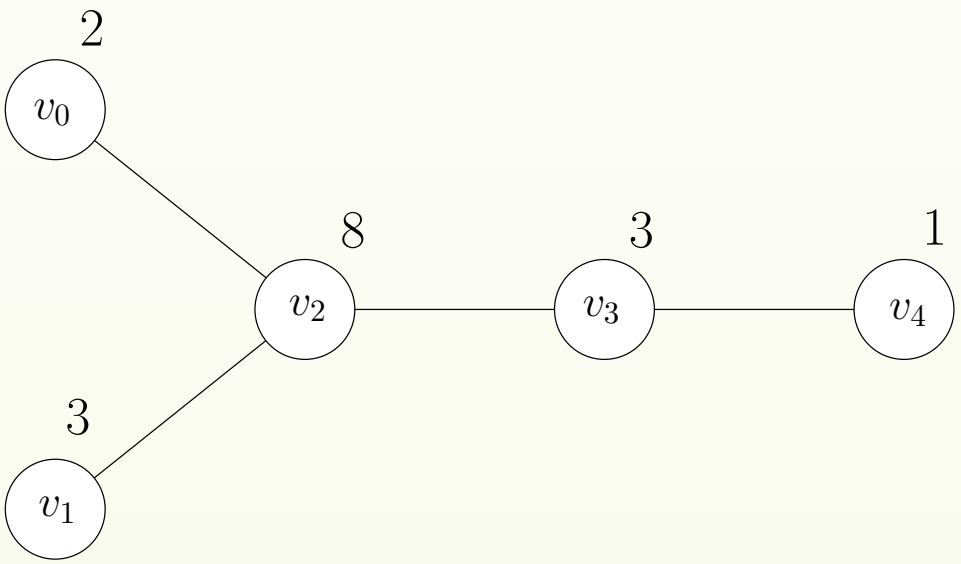
Proof: The set $V' \subseteq V$ is an independent set of weight $W_1 = \sum_{v \in V'} w(v)$ if and only if $V'' = V \setminus V'$ is a vertex cover of weight $W_2 = \sum_{v \notin V'} w(v)$.

Since $W_1 + W_2 = \sum_{v \in V} w(v)$ the statement was proved.

Fix an input graph $G = (V, E)$ with positive vertex weights $w : V \to \mathbb{R}^+$. Let $W = \max\{w(v) \mid v \in V\}$. We build a QUBO matrix of dimension $n = |V|$ such that:

$$Q_{(i,j)} = \begin{cases} -w(v_i), & \text{if } \ i = j, \\ > W, & \text{if } \ i < j \ \text{ and } \ ij \in E. \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

A vertex-weighted graph and its MWIS QUBO matrix.



	v_0	v_1	v_2	v_3	v_4
v_0	-2	0	12	0	0
v_1	0	-3	12	0	0
v_2	0	0	-8	12	0
v_3	0	0	0	-3	12
v_4	0	0	0	0	-1

Theorem

The QUBO formulation given in (4) solves the MWIS Problem.

Proof: Let $\mathbf{x}$ be a Boolean vector corresponding to an optimal solution to the QUBO formulation (4).

Let $D(\mathbf{x}) = \{v_i \mid x_i = 1\}$ be the vertices selected by $\mathbf{x}$.

- If $D(\mathbf{x})$ is an independent set then $-\mathbf{x}^T Q\mathbf{x}$ is its weighted sum.
  For two different solutions $\mathbf{x}_1$ and $\mathbf{x}_2$, which correspond to independent sets, the smallest value of $\mathbf{x}_1^T Q\mathbf{x}_1$ and $\mathbf{x}_2^T Q\mathbf{x}_2$ is better.

- Next assume $D(\mathbf{x})$ is not an independent set and prove that the objective function corresponding to $D(\mathbf{x})$ can be improved.

  Indeed, there must be two vertices v_i and v_j in $D(\mathbf{x})$ such that $v_i v_j$ is an edge in the graph.

  Let $\mathbf{x}_1 = \mathbf{x}$ but set $x_i = 0$, i.e. $D(\mathbf{x}_1) = D(\mathbf{x}) \setminus \{i\}$. We then have

  $$\mathbf{x}_1^T Q \mathbf{x}_1 < \mathbf{x}^T Q \mathbf{x} - W + w(v_i) \leq \mathbf{x}^T Q \mathbf{x}.$$

  v_j.)

  We repeat the process of improving $\mathbf{x}$ until we get an independent set, hence the optimal value of the QUBO holds for some independent set. By the second paragraph of this proof, we know that a maximum weighted independent set corresponds to x^*. See more in [25].

D-Wave architecture is designed to do quantum annealing to solve QUBOs but it is not a complete graph.

In order to "solve" a QUBO problem with the D-Wave machine the logical qubits – the variables in the QUBO formulation – have to be "mapped" onto the physical qubits of the Chimera graph – the graph architecture of the machine –, a process known as "embedding".

Each logical qubit corresponds (via the embedding) to one or more connected physical qubits, called *chain*. *The number of physical qubits is severely limited.*

The *efficiency* of an embedding is measured by the number of physical qubits, the maximum chain and the density of the QUBO matrix.

A *minor embedding* of a graph $G_1 = (V_1, E_1)$ onto a graph $G_2 = (V_2, E_2)$ is a function $f : V_1 \to 2^{V_2}$ that satisfies the following three conditions:

1. The sets of vertices $f(u)$ and $f(v)$ are disjoint for $u \neq v$.
2. For all $v \in V_1$, there is a subset of edges $E' \subseteq E_2$ such that $G' = (f(v), E')$ is connected.
3. If $\{u, v\} \in E_1$, then there exist $u', v' \in V_2$ such that $u' \in f(u)$, $v' \in f(v)$ and $\{u', v'\}$ is an edge in E_2.

For the minor embedding f defined above, G_1 is referred to as the *guest graph* while G_2 is called the *host graph*.

We view a QUBO matrix Q as a weighted adjacency matrix (guest graph) to be embedded onto the D-Wave's Chimera graph (host graph).

The D-Wave quantum computer software package provides a local solver (simulator) option. We can not distribute the software but have provided a small server you can remotely access to use it.
Server domain: cdmtcs.uoa.auckland.ac.nz

User name: CS750
password:

Note that both the user name and password are in capitals.

You can either use ssh to connect to the server (if you are using Mac OS or Linux) or use Putty (if you are using Windows).
D-Wave Systems has recently published a more complicated version of their solver, see the GitHub page if you are interested.

The public_demo folder has several files for demonstration purposes.

You can create your own folder in the home directory, make a copy of the demo files and work in your own folder.

Do not move and modify these files.

There are several files in the public_demo folder:

- C5.alist is the adjacency list for the graph C5 (a cycle of length 5) in the standard adjacency list format (from CS220 or CS320).
- IndSet2QUBO.py is a Python(2) program that generates the QUBO matrix for the Maximum Independent Set Problem for a given graph. The input graph can be either passed to the program using the standard input stream (sys.stdin) or as a command line argument.

- qubo_solver is an exact QUBO solver (not a simulator) developed by Kai Liu. It takes a QUBO matrix from the standard input stream and computes the optimal solution.
- local_solver.py is the main simulator program.
- Several other files generated by the CPLEX package and/or the runtime environment. **Do not change these files.**

We will demonstrate the local simulator and the exact solver in class.

The **graph isomorphism problem** is the computational problem of determining whether two finite graphs are isomorphic.

Instance:	Two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ with $\|V_1\| = \|V_2\|$ and $\|E_1\| = \|E_2\|$.
Question:	Determine whether there exists a bijective edge-invariant vertex mapping (isomorphism) $f : V_1 \to V_2$.

The mapping f is edge-invariant if for every pair of vertices $\{u, v\}$, we have $uv \in E_1$ iff $f(u)f(v) \in E_2$.

This is one of a very few problems in **NP** that is neither known to be solvable in polynomial time nor **NP**-complete.

The problem can be solved in polynomial time for many special classes of graphs and in practice the problem can often be solved efficiently.

In November 2015, L. Babai announced a quasi-polynomial time algorithm for all graphs, which was then presented at STOC 2016.

On 4 January 2017, Babai retracted the quasi-polynomial claim after Harald Helfgott discovered a flaw in the proof.

On 9 January 2017 Babai announced a correction and restored the quasi-polynomial claim, with Helfgott confirming the fix.

The first quantum annealing solution to this problem was given by A. Lucas in 2014.

A *minor embedding* of a graph $G_1 = (V_1, E_1)$ onto a graph $G_2 = (V_2, E_2)$ is a function $f : V_1 \to 2^{V_2}$ that satisfies the following three conditions:

1. The sets of vertices $f(u)$ and $f(v)$ are disjoint for $u \neq v$.
2. For all $v \in V_1$, there is a subset of edges $E' \subseteq E_2$ such that $G' = (f(v), E')$ is connected.
3. If $\{u, v\} \in E_1$, then there exist $u', v' \in V_2$ such that $u' \in f(u)$, $v' \in f(v)$ and $\{u', v'\}$ is an edge in E_2.

For the minor embedding f defined above, G_1 is referred to as the *guest graph* while G_2 is called the *host graph*.

We view a QUBO matrix Q as a weighted adjacency matrix (guest graph) to be embedded onto the D-Wave's Chimera graph (host graph).

Following [18] we present a QUBO objective function F for the graph isomorphism problem which requires n^2 binary variables represented by a binary vector $\mathbf{x} \in \mathbb{Z}_2^{n^2}$:

$$\mathbf{x} = (x_{0,0}, x_{0,1}, \ldots, x_{0,n-1}, x_{1,0}, x_{1,1}, \ldots, x_{1,n-1},$$

$$\ldots, x_{n-1,0}, \ldots, x_{n-1,n-1}).$$

The equality $x_{i,i'} = 1$ encodes the property that the function f maps the vertex v_i in G_1 to the vertex $v_{i'}$ in G_2: $f(v_i) = v_{i'}$.

For this mapping we need to pre-compute n^2 binary constants $e_{i,j}$, $0 \leq i < n$ and $0 \leq j < n$: $e_{i,j} = 1$ if $ij \in E_2$.

The objective function $F(\mathbf{x})$ has the following form:

$$F(\mathbf{x}) = H(\mathbf{x}) + \sum_{ij \in E_1} P_{i,j}(\mathbf{x}), \tag{5}$$

where

$$H(\mathbf{x}) = \sum_{0 \le i < n} \left(1 - \sum_{0 \le i' < n} x_{i,i'} \right)^2 + \sum_{0 \le i' < n} \left(1 - \sum_{0 \le i < n} x_{i,i'} \right)^2, \tag{6}$$

and

$$P_{i,j}(\mathbf{x}) = \sum_{0 \le i' < n} \left(x_{i,i'} \sum_{0 \le j' < n} x_{j,j'} (1 - e_{i',j'}) \right). \tag{7}$$

The function F consists of two parts, $H(\mathbf{x})$ and $\sum_{ij \in E_1} P_{i,j}(\mathbf{x})$.

Each part serves as a penalty for the case when the function f is not an isomorphism.

- The first part H ensures that f is a bijective function:

  $H = 0$ iff the function f encoded by the vector $\mathbf{x}$ is a bijection.
- The second term ensures that f is edge-invariant:

  $\sum_{ij \in E_1} P_{i,j}$ is positive if and only if an edge in E_1 is not preserved under f. Formally, $\sum_{ij \in E_1} P_{i,j}(\mathbf{x}) > 0$ iff there exists an edge $uv \in E_1$ such that

  $f(u)f(v) \notin E_2$.

The objective function $F(\mathbf{x})$ has the following form:

$$F(\mathbf{x}) = H(\mathbf{x}) + \sum_{ij \in E_1} P_{i,j}(\mathbf{x}), \tag{8}$$

where

$$H(\mathbf{x}) = \sum_{0 \le i < n} \left(1 - \sum_{0 \le i' < n} x_{i,i'}\right)^2 + \sum_{0 \le i' < n} \left(1 - \sum_{0 \le i < n} x_{i,i'}\right)^2, \tag{9}$$

and

$$P_{i,j}(\mathbf{x}) = \sum_{0 \le i' < n} \left(x_{i,i'} \sum_{0 \le j' < n} x_{j,j'}(1 - e_{i',j'})\right). \tag{10}$$

Assume that

$$x^* = \min_{\mathbf{x}} F(\mathbf{x}).$$

Then, the mapping f can be 'decoded' from the values of the variables $x_{i,i'}$ using an additional partial function D. Let $\mathcal{F}$ be the set of all bijections between V_1 and V_2.

Then $D : \mathbb{Z}_2^{n^2} \to \mathcal{F}$ is a partial 'decoder' function that re-constructs the vertex mapping f from the vector $\mathbf{x}$, if such f exists.

The domain of D contains all vectors $\mathbf{x} \in \mathbb{Z}_2^{n^2}$ that can be 'decoded' into a bijective function f:

$$\mathrm{dom}(D) = \left\{ \mathbf{x} \in \mathbb{Z}_2^{n^2} \;\middle|\; \sum_{0 \le i' < n} x_{i,i'} = 1, \text{ for all } 0 \le i < n \right.$$
$$\left. \text{and} \sum_{0 \le i < n} x_{i,i'} = 1, \text{ for all } 0 \le i' < n \right\},$$

and

$$D(\mathbf{x}) = \begin{cases} f, & \text{if } \mathbf{x} \in \mathrm{dom}(D), \\ \text{undefined}, & \text{otherwise}, \end{cases}$$

where $f : V_1 \to V_2$ is a bijection such that $f(v_i) = v_{i'}$ iff $x_{i,i'} = 1$.

The following two lemmata will be used to prove correctness of the objective function F in (8).

Lemma

For every $\mathbf{x} \in \mathbb{Z}_2^{n^2}$, $H(\mathbf{x}) = 0$ *iff* $D(\mathbf{x})$ *is defined (in this case* $D(\mathbf{x})$ *is a bijection).*

Lemma

Let $\mathbf{x} \in \mathbb{Z}_2^{n^2}$ *and assume that* $D(\mathbf{x})$ *is a bijective function. Then,* $\sum_{ij \in E_1} P_{i,j}(\mathbf{x}) = 0$ *iff the mapping* $f = D(\mathbf{x})$ *is edge-invariant.*

Theorem

For every $\mathbf{x} \in \mathbb{Z}_2^{n^2}$, $F(\mathbf{x}) = 0$ *iff the mapping* $f : V_1 \to V_2$ *defined by* $f = D(\mathbf{x})$ *is an isomorphism.*

The matrix QUBO can only contain quadratic terms, so some terms of $F(\mathbf{x})$ have to be modified in such a way that this condition is satisfied and the optimal solutions of (**??**) are preserved. To this aim two operations will be performed.

First, any constant term is ignored because removing it does not modify the optimal solutions of (**??**): the value of $F(\mathbf{x})$ is reduced by a constant amount for all $\mathbf{x} \in \mathbb{Z}_2^n$.

Second, as all variables $x_{i,i'}$ are binary, we replace $x_{i,i'}$ with $x_{i,i'}^2$ for all $x_{i,i'}$ with no effect on the value of $F(\mathbf{x})$.

QUBO direct formulation for the graph isomorphism problem: the graph P_3

Consider the path graph P_3 of order 3 and two copies represented as G_1 with edges $E_1 = \{\{0,1\},\{1,2\}\}$ and G_2 with edges $E_2 = \{\{0,1\},\{0,2\}\}$. It is easy to see there are two possible isomorphisms between G_1 and G_2, where we require vertex 1 of G_1 to be mapped to vertex 0 of G_2.

The QUBO formulation requires $3^2 = 9$ variables and the binary variable vector $\mathbf{x} \in \mathbb{Z}_2^9$ is:

$$\mathbf{x} = (x_{0,0}, x_{0,1}, x_{0,2}, x_{1,0}, x_{1,1}, x_{1,2}, x_{2,0}, x_{2,1}, x_{2,2}).$$

By expanding (9) we get:

$$\begin{aligned} H(\mathbf{x}) = &(1 - (x_{0,0} + x_{0,1} + x_{0,2}))^2 + (1 - (x_{1,0} + x_{1,1} + x_{1,2}))^2 \\ &+ (1 - (x_{2,0} + x_{2,1} + x_{2,2}))^2 + (1 - (x_{0,0} + x_{1,0} + x_{2,0}))^2 \\ &+ (1 - (x_{0,1} + x_{1,1} + x_{2,1}))^2 + (1 - (x_{0,2} + x_{1,2} + x_{2,2}))^2 . \end{aligned}$$

Finally we obtain the following penalty terms:

$$P_{0,1} = x_{0,0}x_{1,0} + x_{0,1}(x_{1,1} + x_{1,2}) + x_{0,2}(x_{1,1} + x_{1,2}),$$
$$P_{1,2} = x_{1,0}x_{2,0} + x_{1,1}(x_{2,1} + x_{2,2}) + x_{1,2}(x_{2,1} + x_{2,2}).$$

The objective function $F(\mathbf{x})$ can only contains quadratic terms, so we need to process some of the penalty terms before we can encode them in a QUBO instance. Finally, for all elements $x_{i,i'}$ in $\mathbf{x}$ we map the variable $x_{i,i'}$ to the index $d(x_{i,i'}) = 3i + i' + 1$ (between 1 and 9) and then the entry $Q_{(d(x_{i,i'}),d(x_{j,j'}))}$ is assigned the coefficient of the term $x_{i,i'}x_{j,j'}$ in $F(\mathbf{x})$.

As for each pair $x_{i,i'}$ and $x_{j,j'}$ there are two possible equivalent terms, $x_{i,i'}x_{j,j'}$ and $x_{j,j'}x_{i,i'}$, as a convention, we will use the term that is be mapped to the upper-triangular part of Q.

The upper-triangular matrix representation of Q is:

Table: QUBO matrix for P_3

variables	$x_{0,0}$	$x_{0,1}$	$x_{0,2}$	$x_{1,0}$	$x_{1,1}$	$x_{1,2}$	$x_{2,0}$	$x_{2,1}$	$x_{2,2}$
$x_{0,0}$	-2	2	2	3	0	0	2	0	0
$x_{0,1}$		-2	2	0	3	1	0	2	0
$x_{0,2}$			-2	0	1	3	0	0	2
$x_{1,0}$				-2	2	2	3	0	0
$x_{1,1}$					-2	2	0	3	1
$x_{1,2}$						-2	0	1	3
$x_{2,0}$							-2	2	2
$x_{2,1}$								-2	2
$x_{2,2}$									-2

We have removed a constant value of 6 from $F(\mathbf{x})$ when encoding it into Q, so the value of the optimal solution of $f(\mathbf{x}) = \mathbf{x}^T Q\mathbf{x}$ will be decreased by 6, obtaining the following two optimal solutions:

$$\mathbf{x}_1 = (0,1,0,1,0,0,0,0,1) \text{ and } \mathbf{x}_2 = (0,0,1,1,0,0,0,1,0).$$

Quantum supremacy

The quantum computational advantage for simulating quantum systems was first stated by Feynman in 1981. What is the justification of Feynman's insight? According to the data processing inequality [22], (classical) post-processing cannot increase information. This suggests that to run an accurate classical simulation of a quantum system one must know a lot about the system before the simulation is started. Manin [38] and Feynman [29] have argued that a quantum computer might not need to have so much knowledge.

This line of reasoning seemingly inspired Deutsch [24] to state

> **The postulate of quantum computation**: *Computational devices based on quantum mechanics will be computationally superior compared to digital computers.*

A spectacular support for this postulate came from Shor's 1994 polynomial factoring quantum algorithm [48] in spite of the fact that the problem whether factoring is in **P** was, and still is, open.

In 2011 the syntagm "quantum supremacy" was coined and discussed by J. Preskill in his Rapporteur talk "Quantum Entanglement and Quantum Computing" [46] at the 25th *Solvay Conference on Physics* (Brussels, Belgium, 19–22 October 2011):

> *We therefore hope to hasten the onset of the era of quantum supremacy, when we will be able to perform tasks with controlled quantum systems going beyond what can be achieved with ordinary digital computers.*

What is quantum supremacy?

Recently, quantum supremacy was described in [12] as follows:

> *Quantum supremacy is achieved when a formal computational task is performed with an existing quantum device which cannot be performed using any known algorithm running on an existing classical supercomputer in a reasonable amount of time.*

Why do we need quantum supremacy?

Because the hope sparked by Shor's algorithm (quantum polynomial algorithm for factoring not proved – even after 25 years – to be faster than any competing classical algorithm) and Grover's algorithm (for search an un-ordered data base, proved to be quadratically faster than any classical classical algorithm) didn't come true.

Note the imprecision in the above formulation: the comparison is made with "any known algorithm running on an existing classical supercomputer" and the classical computation takes "a reasonable amount of time". Can this imprecision be decreased or, even better, eliminated?

Quantum supremacy suggests a misleading comparison between classical and quantum computing: if a quantum computer can outdo **any** classical computer on one problem we have quantum supremacy, even if classical computers could be at least as good as quantum ones in solving many (most) other problems. Put it bluntly, *quantum supremacy, if achieved, won't make classical computing obsolete.*

A quantum computational supremacy experiment has to prove both a lower bound and an upper bound.

In Google's proposed experiment [43] the upper bound is given by a quantum algorithm (running on a quantum computer with 49 qubits) sampling from the output distribution of pseudo-random quantum circuits built from a universal gate set—a mathematical fact and an engineering artefact (the construction of the quantum machine)

The lower bound is necessary for proving that no classical computer can simulate the sampling in reasonable time.

Proving lower bounds is notoriously more difficult than demonstrating upper bounds.

Harrow and Montanaro [32] have proposed a reasonable list of criteria for a quantum supremacy experiment:

1. a well-defined computational problem,
2. a quantum algorithm solving the problem which can run on a near-term hardware capable of dealing with noise and imperfections,
3. an amount of computational resources (time/space) allowed to any classical competitor,
4. a small number of well-justified complexity-theoretic assumptions,
5. a verification method that can efficiently distinguish between the performances of the quantum algorithm from **any** classical competitor using the allowed resources.

Is the quest for quantum computational supremacy worthwhile?

Apart publicity and marketing, is the effort of demonstrating the quantum computational supremacy justified? What are the (possible) benefits? Can the claim of quantum computational supremacy be falsified?

The main benefit could be foundational and philosophical: a better understanding of the nature of quantum mechanics through its computational capabilities. A successful quantum supremacy experiment could be a complement to Bell experiment: the later refuted local hidden models of quantum mechanics, while the former *seems* to invalidate the Extended Church-Turing, a foundational principle of classical complexity theory which ensures that the polynomial time class **P** is well defined.

The Thesis places strong constraints, one of them being that *the model of computation is digital*. For example, analog computers are excluded because they assume infinite arithmetic precision. Furthermore, it is known that an infinite precision calculator with operations $+$, x, $=0?$, can factor integers in polynomial time (see [47, 49]).

But, are quantum computers a "reasonable" model of computation? Are quantum systems digital? At first glance quantum computers (and, more generally, quantum systems) appear to be analog devices, since a quantum gate is described by a unitary transformation, specified by complex numbers; a more in-depth analysis is still required.

Is the quest for quantum computational supremacy worthwhile?

What does it take to refute the claim of quantum computational supremacy? This amounts to prove that any computation performed by any quantum computer can be simulated by a classical machine in polynomial time, a weaker form of the Extended Church-Turing Thesis. This statement cannot be proved for the same reasons the Church-Turing Thesis cannot be proved: obviously, they may be disproved.

The paper [44] presents efficient classical boson sampling algorithms and a theoretical analysis of the possibility of scaling boson sampling experiments; it concludes that "near-term quantum supremacy via boson sampling is unlikely".

The proposed experiment is not about solving a problem: it is the computational task of sampling from the output distribution of pseudo-random quantum circuits built from a universal gate set.

This computational task is difficult because as the grid size increases, the *memory needed to store everything increases classically exponentially*. But, do we *really need* to store everything?

The required memory for a $6 \times 4 = 24$–qubit grid is just 268 megabytes, less than the average smartphone, but for a $6 \times 7 = 42$–qubit grid it jumps to 70 terabytes, roughly 10,000 times that of a high-end PC. Google has used Edison, a supercomputer housed by the US National Energy Research Scientific Computing Center and ranked 72 in the Top500 List [1], to simulate the behaviour of the grid of 42 qubits.

The classical simulation stopped at this stage because going to the next size up *was thought to be currently impossible: a 48-qubit grid would require 2,252 petabytes of memory, almost double that of the top supercomputer in the world.* The path to quantum computational supremacy was obvious: if Google could solve the problem with a 50–qubit quantum computer, it would have beaten every other computer in existence.

Simple and clear!

Google was on track to deliver before the end of 2017!

Let us note that many, if not most, discussions about quantum computational supremacy focus on the most exciting possibilities of quantum computers, namely the upper bound.

What about the lower bound? Google's main article on this topic [12] refers cautiously to the lower bound in the abstract:

> *We extend previous results in computational complexity* to argue more formally *that this sampling task must take exponential time in a classical computer.*

They do not claim to have a proof for the lower bound, just a "better formal argument".

Proving the lower bound...

> **Memory assumption**. *Sampling this distribution classically requires a direct numerical simulation of the circuit, with computational cost exponential in the number of qubits.*

The assumption was corroborated by the statement:

> *Storing the state of a 46–qubit system takes nearly a petabyte of memory and is at the limit of the most powerful computers. [43]*

The **Memory assumption** is crucial for the proposed lower bound, and, indeed, this was confirmed very soon. The paper [45] proved that a supercomputer can simulate sampling from random circuits with low depth (layers of gates) of up to 56 qubits.

Better results have been quickly announced, see for example [13]. The limits of classical simulation are not only unknown, but hard to predict.

In spite of this, IBM has announced a prototype of a 50–qubit quantum computer, stating that it "aims to demonstrate capabilities beyond today's classical systems" with quantum systems of this size [2].

According to [3]

> *The successful classical simulation does not undercut the rationale for quantum supremacy experiments. The truth, ironically, is almost the opposite: it being possible to simulate 49–qubit circuits using a classical computer is a precondition for Google's planned quantum supremacy experiment, because it's the only way we know to check such an experiment's results!*

The goal is to get via quantum computing as far as you can up the mountain of exponentiality provided people still see you from the base. Why? Because it's there. "It is not the mountain we conquer but ourselves", as Edmund Hillary aptly said.

Ewin Tang (an 18-year-old undergraduate student at UT Austin) has recently proved [50] that classical computers can solve the "recommendation problem" – given incomplete data on user preferences for products, can one quickly and correctly predict which other products a user will prefer? – with performance comparable to that of a quantum computer.

Is this significant? **Yes**, because in [35] the recommendation problem was hailed as one of the first examples in *quantum machine learning and big data* that would be unlikely to be done classically...

The July 2019 article in Wired magazine which claims that quantum computers could produce true randomness does not even consider the obvious question

> *Are the quantum random bits produce by the quantum computer* **provably better** *than pseudo-random bits?*

In the spirit of "quantum supremacy" they claim a watered down advantage:

> *For a classical computer, the task [performed by the quantum algorithm generating the random bits] becomes exponentially harder as the number of bits in the string gets larger.*

Two lessons

- Do not to underestimate the importance of mathematical modelling and proving (lower bounds, in particular).
- The conversation on quantum computing, quantum cryptography and their applications needs an infusion of modesty (if not humility), more technical understanding and clarity as well as less hype. Raising false expectations could be harmful for the field.

The race continues! See more in [15].

Supplementary material

Let $s_1 \cdots s_n$ be a binary string. A monochromatic arithmetic progression of length k is a substring

$$s_i s_{i+t} s_{i+2t} \cdots s_{i+(k-1)t},$$

$1 \leq i \leq i + (k-1)t \leq n$ with all characters equal (0 or 1) for some $t > 0$.

The strings

$$\mathbf{0}110\mathbf{0}110\mathbf{0} \text{ and } 011\mathbf{0}0\mathbf{1}101$$

have each a monochromatic arithmetic progression of length 3: 1, 5, 9 for 0 and 3, 6, 9 for 1.

In fact, one can prove that

all 512 binary strings of length 9 have a monochromatic arithmetic progression of length 3.

Van der Waerden finite theorem

For every natural k there is a natural $n > k$ such that every string of length n contains a monochromatic arithmetic progression of length k.

▸ VDW

Proof

Let C_i be a computably enumeration of all prefix-free TMs and construct the prefix-free TM U by

$$U(1^i 0x) = C_i(x).$$

▸ USTheorem

Proof

To prove that $H(x^*) \geq |x^*| - c$ we construct the prefix-free TM

$$D(p) = U(U(p))$$

and pick the constant c coming from the universality theorem (applied to U and D). Take $x = y^*$, $z = x^*$. One has:

$$D(z) = U(U(z)) = U(U(x^*)) = U(x) = U(y^*) = y,$$

so

$$H_D(y) \leq |z| = |x^*| = H(x),$$

$$|x| = |y^*| = H(y) \leq H_D(y) + c \leq H(x) + c.$$

▸ StringComplexity

Proof

Given $\omega_1\omega_2\cdots\omega_n$ we run in parallel $U(p)$ on various programs till we get enough halting programs $p_1,\ldots,p_N$ such that

$$\sum_{i=1}^{N} 2^{-|p_i|} \geq 0.\omega_1\omega_2\cdots\omega_n.$$

Every program q with $|q| \leq n$, different from all p_i's above, doesn't stop as otherwise

$$\Omega_U < 0.\omega_1\omega_2\cdots\omega_n + 2^{-n} \leq \sum_{i=1}^{N} 2^{-|p_i|} + 2^{-|q|} \leq \Omega_U,$$

a contradiction. ▸ OmegaHaltTheorem

Proof

If $S_{\mathbf{x}}$ contains an infinite c.e. (equivalently, computable) subset B, then the p.c. function $\varphi(i) = 1$ for $i \in B$ has the property that $\varphi(i) = x_i$, for every $i \in B$.

Conversely, if there exists a p.c. function φ with infinite domain such that for every $i \in \mathrm{dom}(\varphi)$ we have $x_i = \varphi(i)$, then either

$\{i \in \mathrm{dom}(\varphi) \mid \varphi(i) = 1\}$ is an infinite c.e. subset of $S_{\mathbf{x}}$ or

$\{i \in \mathrm{dom}(\varphi) \mid \varphi(i) = 0\}$ is an infinite c.e. subset of the complement of $S_{\mathbf{x}}$.

▸ bi-immunity

Proof

Define M as follows: on input x compute $y = U(x)$ and the smallest number of programs $p_1, \ldots, p_t \in \mathrm{dom}(U)$ (if they exist) with

$$\sum_{i=1}^{t} 2^{-|p_i|} \geq 0.y.$$

Let $M(x)$ be the first (in quasi-lexicographical order) string not belonging to the set $\{U(p_1), U(p_2), \ldots, U(p_t)\}$ if both y and t exist, and $M(x) = \infty$ if $U(x) = \infty$ or t does not exist.

If $M(x) < \infty$ and x' is a string with $U(x) = U(x')$, then $M(x) = M(x')$.

Applying this to an arbitrary x with $M(x) < \infty$ and the program $x' = (U(x))^*$ yields

$$H_M(M(x)) \leq |x'| = H_U(U(x)). \tag{11}$$

Furthermore, by the universality of U there is a constant $c > 0$ with

$$H_U(M(x)) \leq H_M(M(x)) + c, \tag{12}$$

for all x with $M(x) < \infty$.

Take $U(x) = \omega_1 \cdots \omega_n$. From the construction of M (and the fact that Omega solves the halting problem) we conclude that $H_U(M(x)) \geq n$. Using (12) and (11) we obtain

$$\begin{aligned} n &\leq H_U(M(x)) \\ &\leq H_M(M(x)) + c \\ &\leq H_U(U(x)) + c \\ &= H_U(\omega_1\omega_2 \cdots \omega_n) + c. \end{aligned}$$

▸ OmegaMaxComplTheorem

Proof

The sequence of rationals $r_N = \sum_{i=1}^{N} 2^{-|f(i)|}$, where f is a computable enumeration of the domain of U, is computable, increasing and converges to Ω.

▸ CEOmegaTheorem

Van der Waerden theorem

For any positive integers k there is a positive integer γ such that every bit string of length more than γ contains an arithmetic progression with k occurrences of the same digit or colour, i.e. a monochromatic arithmetic progression of length k.

▸ SpuriousCorrelations

The CBS drama TV show Numb3rs, http://www.cbs.com/primetime/numb3rs/, season 5; episode 5; scene 6) includes the following dialogue:

> LARRY: *Ah, Charles, my ambulatory reference book. Chaitin's Omega Constant...?*
>
> CHARLIE: *Omega equals .00787499699. Why, what're you working on?* (sees the file, reacts) *Oh. FBI file.*

The math is explained at http://numb3rs.wolfram.com/505.

▸ Incomputability of Omega

References

[1] Edison supercomputer in TOP 500 ranking.
https://www.top500.org/list/2017/06/?page=1, June 2017.

[2] IBM builds 50-qubit quantum computer.
http://techvibesnow.com/ibm-builds-50-qubit-quantum-computer/, November 2017.

[3] S. Aaronson.
Shtetl-Optimized – 2^n is exponential, but 2^{50} is finite.
https://www.scottaaronson.com/blog/?p=3512, November, 12 2017.

References (cont.)

[4] A. A. Abbott.
The Deutsch-Jozsa problem: De-quantisation and entanglement.
Natural Computing, 11(1):3–11, 2011.

[5] A. A. Abbott and C. S. Calude.
Understanding the quantum computational speed-up via de-quantisation.
Electronic Proceedings in Theoretical Computer Science, 26:1–12, 2010.

[6] A. A. Abbott, C. S. Calude, J. Conder, and K. Svozil.
Strong Kochen-Specker theorem and incomputability of quantum randomness.
Physical Review A, 86(062109), Dec 2012.

References (cont.)

[7] A. A. Abbott, C. S. Calude, M. J. Dinneen, and N. Huang.
Experimentally probing the algorithmic randomness and incomputability of quantum randomness.
Physica Scripta, 94(4):045103, feb 2019.

[8] A. A. Abbott, C. S. Calude, and K. Svozil.
Value indefiniteness is almost everywhere.
Physical Review A, 89(3):032109–032116, 2014.

[9] A. A. Abbott, C. S. Calude, and K. Svozil.
A non-probabilistic model of relativised predictability in physics.
Information, 6(4):773–789, 2015.

References (cont.)

[10] D. Aharonov, W. v. Dam, J. Kempe, Z. Landau, S. Lloyd, and O. Regev.
Adiabatic quantum computation is equivalent to standard quantum computation.
arXiv:quant-ph/0405098, March 2005.

[11] E. H. Allen and C. S. Calude.
Quassical computing.
https://arXiv:1805.03306v1, May 2018.

[12] S. Boixo, S. V. Isakov, V. N. Smelyanskiy, R. Babbush, N. Ding, Z. Jiang, M. J. Bremner, J. M. Martinis, and H. Neven.
Characterizing quantum supremacy in near-term devices.
arXiv:1608.00263 [quant-ph], April 2017.

References (cont.)

[13] S. Boixo, S. V. Isakov, V. N. Smelyanskiy, and H. Neven.
Simulation of low-depth quantum circuits as complex undirected graphical models.
https://arxiv.org/pdf/1712.05384.pdf, January 2018.

[14] C. Calude.
Information and Randomness—An Algorithmic Perspective.
Springer, Berlin, second edition, 2002.

[15] C. Calude and E. Calude.
The road to quantum computational supremacy.
http://arxiv.org/abs/1712.01356v2, November 2017.

[16] C. S. Calude.
De-quantizing the solution of Deutsch's problem.
International Journal of Quantum Information, 5(3):409–415, Jun 2007.

References (cont.)

[17] C. S. Calude, E. Calude, and M. J. Dinneen.
Adiabatic quantum computing challenges.
ACM SIGACT News, 46(1):40–61, March 2015.

[18] C. S. Calude, M. J. Dinneen, and R. Hua.
Qubo formulations for the graph isomorphism problem and related problems.
Theoretical Computer Science, 2017,
https://doi.org/10.1016/j.tcs.2017.04.016.

[19] C. S. Calude and N. J. Hay.
Every computably enumerable random real is provably computably enumerable random.
Log. J. IGPL, 17(4):351–374, 2009.

References (cont.)

[20] C. S. Calude and G. Longo.
The deluge of spurious correlations in big data.
Foundations of Science, pages 1–18, 2016.

[21] C. S. Calude and G. Păun.
Computing with Cells and Atoms.
Taylor & Francis Group, London and New York, 2001.

[22] T. M. Cover and J. A. Thomas.
Elements of Information Theory.
John Wiley & Sons, New York, 1991.

[23] D-Wave Systems, `www.dwavesys.com`.
D-Wave Problem-Solving Handbook. User Manual, March 2018.

References (cont.)

[24] D. Deutsch.
Quantum theory, the Church-Turing principle and the universal quantum computer.
Proceedings of the Royal Society of London Series A, 400:97–117, Jan 1985.

[25] M. J. Dinneen and R. Hua.
Dominating Set and Edge Cover Using D-Wave Machines.
Presentation, 32 pp., December 2016.

[26] R. Downey and D. Hirschfeldt.
Algorithmic Randomness and Complexity.
Springer, Berlin, 2010.

References (cont.)

[27] A. Einstein, B. Podolsky, and N. Rosen.
Can quantum-mechanical description of physical reality be considered complete?
Physical Review, 47(10):777–780, May 1935.

[28] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser.
Quantum computation by adiabatic evolution.
arXiv:quant-ph/0001106, January 2000.

[29] R. P. Feynman.
Simulating physics with computers.
International Journal of Theoretical Physics, 21:467–488, 1982.

References (cont.)

[30] R. Graham.
Some of my favorite problems in Ramsey Theory.
INTEGERS, The Electronic Journal of Combinatorial Number Theory, 7(2):#A2, 2007.

[31] J. Gruska.
Quantum Computing.
McGraw-Hill, London, 1999.

[32] A. W. Harrow and A. Montanaro.
Quantum computational supremacy.
Nature, 549(7671):203–209, 09 2017.

[33] N. Johansson and J.-Å. Larsson.
Efficient classical simulation of the Deutsch–Jozsa and Simon's algorithms.
Quantum Information Processing, 16(9):233, Aug 2017.

References (cont.)

[34] M. Kac.
What is random?
American Scientist, 71:405–406, 1983.

[35] I. Kerenidis and A. Prakash.
Quantum recommendation system.
In C. H. Papadimitrou, editor, *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*, pages 49:1–49:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany, 2017.

[36] A. Kulikov, M. Jerger, A. Potočnik, A. Wallraff, and A. Fedorov.
Realization of a quantum random generator certified with the Kochen-Specker theorem.
Phys. Rev. Lett., 119:240501, Dec 2017.

References (cont.)

[37] A. K. Lenstra, J. P. Hughes, M. Augier, J. W. Bos, T. Kleinjung, and C. Wachter.
Ron was wrong, Whit is right.
Santa Barbara: IACR: 17, https://eprint.iacr.org/2012/064.pdf, 2012.

[38] Y. I. Manin.
Vychislimoe i nevychislimoe [Computable and Noncomputable] (in Russian). Sov. Radio. pp. 13–15. (Checked 30 November 2017).
http://www.worldcat.org/title/vychislimoe-i-nevychislimoe/oclc/11674220, 1980.

References (cont.)

[39] C. McGeoch.
Adiabatic Quantum Computation and Quantum Annealing. Theory and Practice.
Morgan & Claypool Publishers, 2014.

[40] C. C. McGeoch, R. Harris, S. P. Reinhardt, and P. I. Bunyk.
Practical annealing-based quantum computing.
Computer , *www. computer. org/ computer* , pages 38–46, 2019.

[41] D. N. Mermin.
What's wrong with these elements of reality?
Physics Today, 43(6):9–10, June 1990.

[42] D. N. Mermin.
Quantum Computer Science.
Cambridge University Press, Cambridge, 2007.

References (cont.)

[43] C. Neill, P. Roushan, K. Kechedzhi, S. Boixo, S. V. Isakov, V. Smelyanskiy, R. Barends, B. Burkett, Y. Chen, Z. Chen, B. Chiaro, A. Dunsworth, A. Fowler, B. Foxen, R. Graff, E. Jeffrey, J. Kelly, E. Lucero, A. Megrant, J. Mutus, M. Neeley, C. Quintana, D. Sank, A. Vainsencher, J. Wenner, T. C. White, H. Neven, and J. M. Martinis.
A blueprint for demonstrating quantum supremacy with superconducting qubits.
arXiv:1709.06678 [quant-ph].

[44] A. Neville, C. Sparrow, R. Clifford, E. Johnston, P. M. Birchall, A. Montanaro, A. L. A. Neville, C. Sparrow, R. Clifford, E. Johnston, P. M. Birchall1, A. Montanaro4, and A. Laing.
No imminent quantum supremacy by boson sampling.

References (cont.)

https://arxiv.org/pdf/1705.00686.pdf, May 2017.

[45] E. Pednault, J. A. Gunnels, G. Nannicini, L. Horesh, T. Magerlein, E. Solomonik, and R. Wisnieff.
Breaking the 49-qubit barrier in the simulation of quantum circuits.
https://arxiv.org/abs/1710.05867, October 2017.

[46] J. Preskill.
Quantum computing and the entanglement frontier.
In H. M. Gross, D. and A. Sevrin, editors, *The Theory of the Quantum World*, pages 63–80, Singapore, November 10 2012. World Scientific Publishing.
arXiv:1203.5813 [quant-ph].

References (cont.)

[47] A. Shamir.
Factoring numbers in O(log n) arithmetic steps.
Information Processing Letters, 8(1):28–31, 1979.

[48] P. W. Shor.
Algorithms for quantum computation: discrete logarithms and factoring.
In *Proceedings of the 35th Annual Symposium of on Foundations of Computer Science, Santa Fe, NM, Nov. 20-22, 1994*. IEEE Computer Society Press, November 1994.
arXiv:quant-ph/9508027.

References (cont.)

[49] V. Tamma.
Analogue algorithm for parallel factorization of an exponential number of large integers: Ii—optical implementation.
Quantum Information Processing, 15(12):5243–5257, Dec 2016.

[50] E. Tang.
A quantum-inspired classical algorithm for recommendation systems.
https://arxiv.org/pdf/1807.04271.pdf, July 2018.