Quantum algorithms

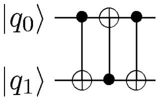
Quantum computing

G. Chênevert

Jan. 27, 2023

Review exercise

What is the effect of 3 consecutive CNOT gates like below?



Answer: a SWAP gate

Quantum algorithms

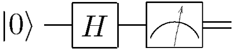
First quantum algorithms

Quantum complexity

Grover's algorithm

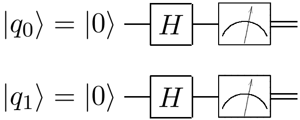
True random number generator

On 1 bit:



outputs 0 or 1 with probability $\frac{1}{2}$

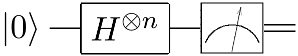
On 2 bits:



outputs 0, 1, 2 or 3 with probability $\frac{1}{4}$

True random number generator

In general



ouputs any integer in $[\![0, 2^n[\![$ with probability $\dfrac{1}{2^n}$.

$$\underbrace{H|0\rangle \otimes \cdots \otimes H|0\rangle}_{n} = \frac{1}{2^{n/2}} \sum_{x<2^n} |x\rangle$$

Example : the Deutsh algorithm

There exists 4 boolean functions f of a single variable: two of them are constant

$$f(x) \equiv 0, \qquad f(x) \equiv 1, \qquad \text{(type 0)}$$

while the two others are not:

$$f(x) = x, \qquad f(x) = 1 \oplus x. \qquad \text{(type 1)}$$

Example : the Deutsch algorithm

Suppose you are given one of those four functions f (as a black box: the only thing you can do is evaluate the function on inputs of your choice) and asked what type it is.

In the classical world, clearly two evaluations of f are needed (and sufficient).

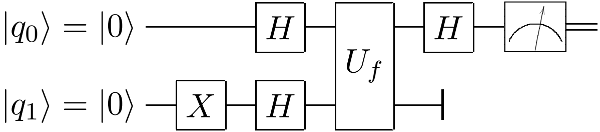
`input` f

`output` $f(0) \oplus f(1)$

The Deutsch algorithm finds out the type of f with *a single quantum evaluation*.

Example : the Deutsch algorithm

The following circuit computes the type of f:



Proof: Exercise!

Quantum algorithms

First quantum algorithms

Quantum complexity

Grover's algorithm

Faster computations

The whole point of quantum computing is that some quantum algorithms exhibit **quantum advantage**: *i.e.* run "faster" than the best known classical algorithms

In extreme cases, this leads to

quantum supremacy: *i.e.* the ability to compute things that could never practically be achieved with classical computers.

2019: Sampling random quantum circuits on 53 qubits (Google) supremacy?

2020, 2021, 2022: Quantum computational advantage using photons (USTC)

Complexity of an algorithm

Classically: the complexity of an algorithm $\mathcal{A}$ is a bound on the number of computing steps needed for an input of a given size

i.e. the function

$$n \mapsto \max_{|x|=n} N_{\mathcal{A}}(x)$$

where $N_{\mathcal{A}}(x)$ denotes the number of steps used to perform $\mathcal{A}$ on x in a given computing model (usually: Turing machines)

Quantum computing model

There exist a (rather inconvenient) notion of quantum Turing machine

Most people prefer to use the (equivalent) **quantum circuit model**:

- given input x: a circuit $U_{\mathcal{A},x}$ made out of quantum gates taken from a standard generating set is prepared
- the output y is the result of the measure of $U_{\mathcal{A},x}|0\rangle$
- (then some classical post-processing may be applied)

The **complexity** of the quantum algorithm is a bound on the number of gates needed:

$$n \mapsto \max_{|x|=n} |U_{\mathcal{A},x}|.$$

The Deutsch-Jozsa problem

Given a boolean function $f : \{0,1\}^n \to \{0,1\}$ assumed to be either

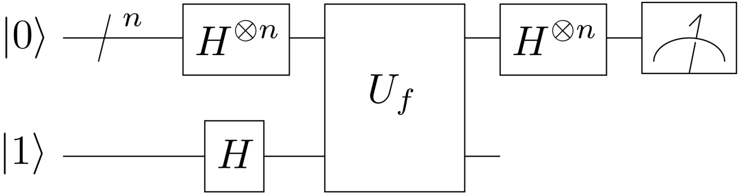
- constant: $f(x) \equiv 0$ or $f(x) \equiv 1$ ("type 0") or
- balanced: $|\{x \mid f(x) = 0\}| = |\{x \mid f(x) = 1\}| = 2^{n-1}$ ("type 1"),

problem: compute its type.

A classical algorithm needs at least $2^{n-1} + 1$ evaluations of f to decide

$\Longrightarrow$ **EXP** (EXPonential time) complexity class

The Deutsch-Jozsa algorithm



Proposition: output is $|0\rangle^{\otimes n} \iff f$ is constant

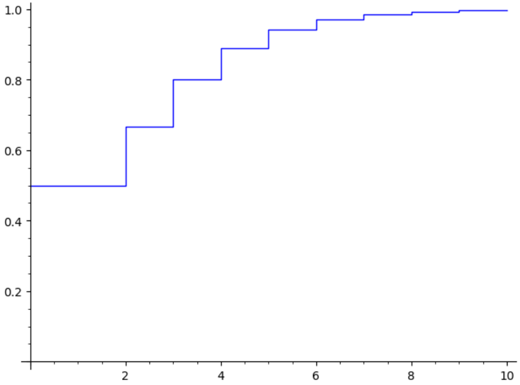
EQP (Exact Quantum Polynomial time) complexity class

$\implies$ exponential speedup !

Deutsch-Jozsa: remarks

- Here U_f is considered an **oracle** for f (black box implementation)
- Classical decision algorithm: $2^{n-1}+1$ evaluations are required to guarantee the answer... but we can get a probable answer with much less evaluations.
- With k evaluations, assuming constant and balanced functions are equiprobable:
  - if not all values are equal, f is certainly balanced
  - if all values are equal, f is constant with probability (Bayes)

$$\frac{1}{1+\frac{1}{2^{k-1}}}$$



How many evaluations are needed to be 99 % sure whether f is constant or balanced ?

Answer: $k \geq 1 + \log_2\left(\frac{1}{\frac{1}{0.99}-1}\right) \approx 7.63$ so 8 evaluations would be enough in practice

Probabilistic algorithms

In practice: we prefer a fast algorithm with a good probability of giving a right answer to a slow algorithm that is always right!

Example: Rabin-Miller *vs* AKS primality tests

The Deutsch-Jozsa problem is in the (classical)

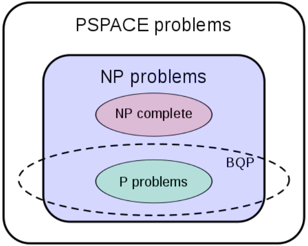
BPP (Bounded-error Probabilistic Polynomial time) complexity class

But since quantum algorithms are typically also probabilistic...

The speedup here is not so impressive after all!

Aside: the Complexity Zoo

- We know that $\mathbf{P} \subseteq \mathbf{NP}$ (Nondeterministic Polynomial time)
- Whether the inclusion is strict is an open question ($1,000,000)
- We also know that $\mathbf{P} \subseteq \mathbf{BPP} \subseteq \mathbf{BQP}$ (Bounded-error Quantum Polynomial time)
- Reverse inclusions *also unknown*; complexity classes are a real zoo



Algorithms with quantum advantage

Two algorithms displaying a clear quantum advantage over the classical counterparts:

- **Grover**'s algorithm

  unstructured search among n items in $\mathcal{O}(\sqrt{n})$

- **Shor**'s algorithm

  factoring a n-bit integer in $\mathcal{O}(n^3 \log n)$

Quantum algorithms

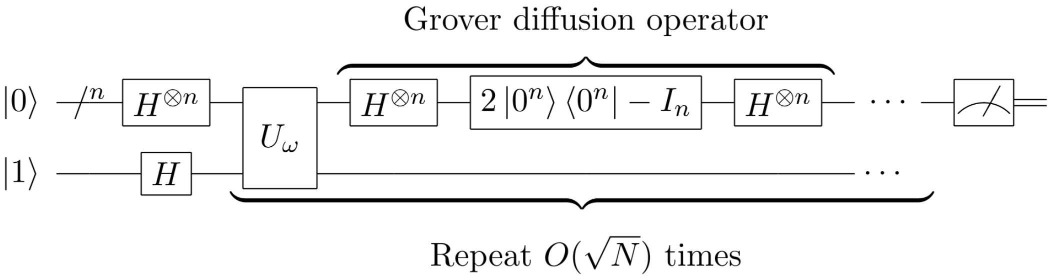
First quantum algorithms

Quantum complexity

Grover's algorithm

Grover (1970)

Grover (1996)


Grover diffusion operator
|0⟩
n
H⊗n
Uω
H⊗n
2 |0n⟩ ⟨0n| − In
H⊗n
|1⟩
H
Repeat O(√N) times

Search problem

Suppose we have a decision function $f : X \to \{0, 1\}$ defined on a set X of size N.

The search problem defined by f is to find some $x \in X$ for which $f(x) = 1$.

Examples: database queries, factoring integers, bitcoin mining, ...

In the general (unstructured) case: a classical algorithm requires $\mathcal{O}(N)$ queries.

(Of course can do better if *e.g.* the data is sorted)

Grover's algorithm

Performs unstructured searches for arbitrary criteria in $\mathcal{O}(\sqrt{N})$ time.

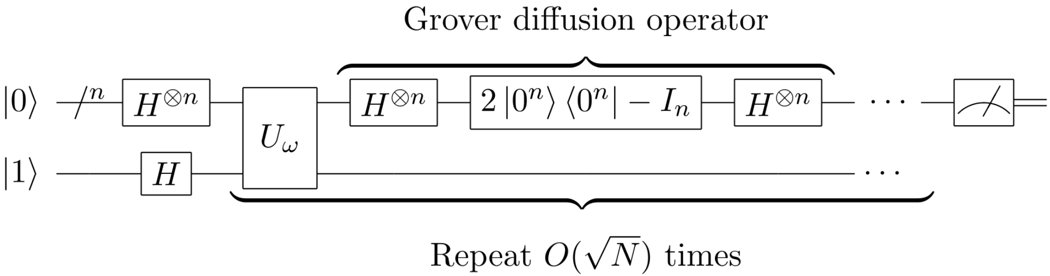
$\implies$ **quadratic speedup**

Works in two steps:

- phase inversion
- amplitude amplification

iterated a certain number of times

Circuit for Grover's algorithm



Phase inversion

Simplifying assumptions:

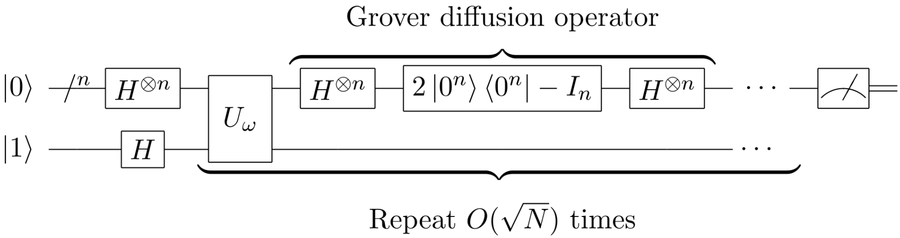
- $X = [\![0, N[\![$
- $N = 2^n$
- the equation $f(x) = 1$ admits a unique solution $\omega \in X$

So the problem is now: find $\omega \in X$ given access to a oracle for $f : [\![0, N[\![\to \{0, 1\}$

$$\text{where} \quad f(x) = \begin{cases} 1 & \text{if } x = \omega \\ 0 & \text{else.} \end{cases}$$

Phase inversion

ω is detected by inverting its phase: "$U_\omega|x\rangle \ = (-1)^{f(x)}|x\rangle$"



Actually

$$U_\omega\,|x\rangle \otimes |-\rangle = (-1)^{f(x)}\,|x\rangle \otimes |-\rangle$$

This is exactly what the oracle U_f does! So in fact "$U_\omega = U_f$".

Amplitude amplification

The **Grover diffusion operator** G is

$$G = 2|s\rangle\langle s| - I$$

where

$$|s\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle.$$

Geometrical interpretation:

$$G|s\rangle = |s\rangle$$

$$G|\psi\rangle = -|\psi\rangle \qquad \text{when } \langle s|\psi\rangle = 0$$

$U_s = -G$ is a reflection through the hyperplane normal to $|s\rangle$

Amplitude amplification

Remark: U_ω is a reflection, too.

Actually U_ω acts on $\mathcal{V} = \mathcal{V}_N \otimes |-\rangle$ as

$$I - 2|\omega\rangle\langle\omega| = \mathrm{diag}(1, \ldots, \underbrace{-1}_{\omega}, \ldots, 1).$$

In general $I - 2|\psi\rangle\langle\psi|$ is a reflection through the hyperplane normal to $|\psi\rangle$.

GU_ω: unitary transformation of $\mathcal{V}$ that inverts every vector $|\psi\rangle$ orthogonal to both $|s\rangle$ and $|\omega\rangle$ – and acts as a rotation in the plan spanned by $|s\rangle$ and $|\omega\rangle$

Amplitude amplification

Consider unitary $|s'\rangle \sim |s\rangle - \langle\omega|s\rangle|\omega\rangle$, and write $\langle\omega|s\rangle = \frac{1}{\sqrt{N}} = \sin\frac{\theta}{2}$.

Initial state:

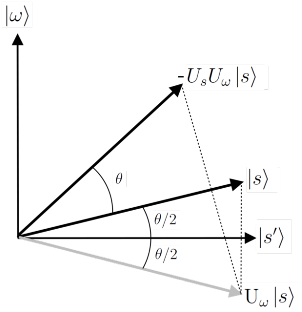
$$|\psi\rangle = |s\rangle = \cos\tfrac{\theta}{2}\,|s'\rangle + \sin\tfrac{\theta}{2}\,|\omega\rangle$$

GU_ω is a rotation of θ (exercise!), so after k iterations:

$$(GU_\omega)^k|\psi\rangle = \cos(\tfrac{\theta}{2} + k\theta)\,|s'\rangle + \sin(\tfrac{\theta}{2} + k\theta)\,|\omega\rangle$$

$$\mathbb{P}\big[\,\mathcal{M}(GU_\omega)^k|\psi\rangle = |\omega\rangle\,\big] = \sin^2(\tfrac{\theta}{2} + k\theta)$$

Optimal number of iterations



Each iteration brings the state closer to $|\omega\rangle$ by an angle of $\theta = 2\arcsin\frac{1}{\sqrt{N}}$.

Until it starts moving away... Sage visualization

Optimal number of iterations

So, in order to maximize the probability of measuring $|\omega\rangle$, take

$$(k+\tfrac{1}{2})\theta \approx \frac{\pi}{2} \qquad \iff \qquad k \approx \frac{\pi}{2\theta} - \frac{1}{2}$$

When N is large (interesting case!) we have $\theta \approx \sin\theta = \dfrac{2}{\sqrt{N}}$

so the optimal number of iterations is $k \approx \dfrac{\pi\sqrt{N}}{4}$.

Closely related to this rather surprising way to approximate π !

Implementation of G

$$G = 2|s\rangle\langle s| - I$$

- G is more easily computed if we change the basis:

$$G = H^{\otimes n} \otimes \underbrace{(2|0\rangle\langle 0| - I)}_{G_0} \otimes H^{\otimes n}$$

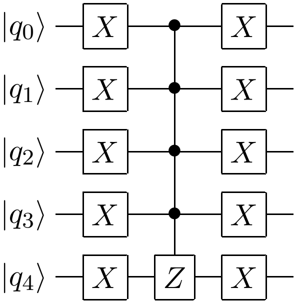
- $G_0 \sim -G_0 = U_0 = \mathrm{diag}(-1, 1, \ldots, 1)$:

$$U_0|x\rangle = \begin{cases} -|x\rangle & \text{if } x = 0 \\ |x\rangle & \text{if } x \neq 0. \end{cases}$$

Implementation of G

Example: with $n = 5$

G_0:



G: