VIII. Quantum Complexity Theory

Aim of complexity theory: understand & classify difficulty of problems.

Difficulty typ. measured in terms of resources needed (time, memory) to solve problem, as function of the problem size n (= # bits needed to specify the specific instance of the problem, i.e., the input).

Focus on decision problems (= yes/no problems):

Problem: $f : \{0,1\}^n \longrightarrow \{0,1\}$

$f : \underbrace{x}_{\text{instance}} \longmapsto f(x) = \text{yes/no}$

Note: Non-decision problems can typ. be rephrased as a few calls to a decision problem.

CS terminology: $L := \{ x \in \underbrace{\{0,1\}^*}_{\text{any length}} \mid f(x) = 1 \}$ is called language.

i.e.: Computing $f(x) \Leftrightarrow$ deciding if $x \in L$.

1. Classical complexity classes

Which comp. model? — Church-Turing thesis:
all comp. models equiv. w/ poly overhead.
→ equiv. for our purposes.

Class P ("polynomial time"):

$f(x)$ can be computed in time $(=\#$ of operations$)$
$\text{poly}(|x|)$
↳ length of x

(Those are commonly considered efficiently solvable.)

Examples in P:

* multiplication, addition, ... (or decision versions)
* primality testing
* gcd

(Note: We can also define other time classes, e.g. EXP)

NP ("non-deterministic polynomial time")

Problem can be solved in time poly($|x|$) by a "non-deterministic computer", i.e. which can check many inputs y in parallel.

Equiv.: If $f(x) = 1$, there exists a proof (witness) y which can be checked in time poly($|x|$) with a verifier $v(x, y) = 1$ (and only if $f(x) = 1$):

I.e.: $\exists\, v(x, y)$ s.th.

$x \in L \Rightarrow \exists y : v(x, y) = 1$ ("proof accepted")

$x \notin L \Rightarrow \forall y \; v(x, y) = 0$ ("proof rejected")

Typ., y is the 'solution' to the problem.

Examples:

* Graph coloring (color graph w/out eq. adjacent colors)

"yes" ≡ coloring exists ⇒ proof = a valid coloring

"no" ≡ no coloring exists ⇒ no valid proof

* k-SAT: variables $x_1, \ldots, x_k$;

"clause" $g_j = x_{a_j} \vee \bar{x}_{b_j} \vee x_{c_j}$ etc.

(i.e.: each var is x_i or $\bar{x}_i$, and g_j = OR of 3 variables)

Question: Does there exist a satisfying assignment $x_1, \dots, x_k$ s.t. $g_j = 1 \ \forall j$?

Yes instance: Proof $= (x_1, \dots, x_k)$

No instance: no proof exists

* (Prime) factor decomposition
* graph isomorphism: are two graphs isomorphic?

Can problems in NP be arbitrarily hard?

→ No, e.g. games are typ. not NP (due to their branching structure).

→ NP contains "reasonably hard" problems.

→ Generally (?) believed: $P \neq NP$ (note: $P \subset NP$).

Classes beyond NP:

PSPACE (polynomial space):

Problems which can be solved using only $\text{poly}(|x|)$ memory (no time limit, but note that time $\leq \exp(\text{space})$)

If $P \neq NP$: The "hardest" problems in NP should be interesting class of hard problems.

How can we identify "hardest" problems in NP?

NP-completeness: A problem is NP-complete if we can map ("reduce") any problem in NP to it in poly time.
(i.e.: If we have a way to solve an NP-complete problem in poly time, we can solve any NP-problem in poly time)

Examples for NP-complete problems

* graph coloring, k-SAT for $k \geq 3$

Non-complete (NP-intermediate) problems:

* factoring, graph isomorphism

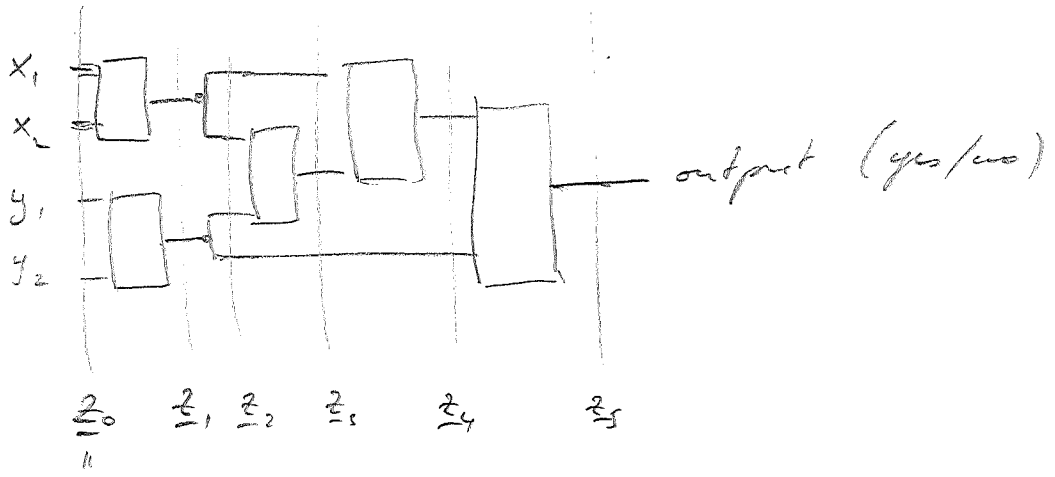
The Cook-Levin-theorem: 3-SAT is NP-complete

Aim: Reduce any NP-problem to 3-SAT.

General NP-problem: Given by verifier $v(x, y)$,

yes instance $\Longleftrightarrow \exists\, y : v(x, y) = 1$.

$v(x,y) \hat{=}$ circuit:



(x_1, x_2, y_1, y_2), etc.

"history state" $(\vec{z}_0, \vec{z}_1, \ldots, \vec{z}_5)$ of verifier checking proof y

Construct 3-SAT instance which ensures $(\vec{z}_0, \vec{z}_1, \ldots)$ is a <u>valid history of v</u> for instance x and <u>some</u> proof y!

* for each input x_i, add clause which ensures that

$$(z_0)_i = x_i. \quad (\text{i.e., } g_i = x_i \text{ or } g_i = \bar{x}_i)$$

* for each <u>gate</u> (incl. copy), add clauses which ensure that the gate is done correctly (i.e. possible input/output) – acts on 3 bits each

* Add a clause requiring the output z_5 to be "yes".

$\Rightarrow$ 3-SAT instance

$$\exists y : v(x,y) = 1 \iff \text{3-SAT satisfiable.}$$

$\Rightarrow$ 3-SAT NP-complete!

Central idea of proof: Construct time-history of verifier V and write k-SAT problem for it.

Note: This k-SAT corresponds to a classical local Hamiltonian on a 2D lattice $\Rightarrow$ finding Ground States of 2D lattices is NP-complete.

(Note: NP-completeness does not tell us about average case complexity — many hard problems are only hard in certain param. regimes.)

Quantum Complexity Classes:

BQP ("bounded-error quantum polynomial time")

The class of problems which can be solved by a (circuit-based) quantum computer in time poly(|x|) with bounded error.

Note: Bounded error $\Leftrightarrow$ yes-instance: $p(\text{output}=1) \geq 2/3$

no-instance: $p(\text{output}=1) < 1/3$.

We can use <u>amplification</u> (i.e., many algorithm poly # of times and using majority vote) to get this up to

$$p(\text{output} = 1 \mid \text{yes}) \geq 1 - 2^{-|x|}$$

$$p(\text{output} = 1 \mid \text{no}) \leq 2^{-|x|},$$

similarly, $p(\text{output} = 1 \mid \text{yes}) \geq \frac{1}{2} + \frac{1}{\text{poly}(|x|)}$

$$p(\text{output} = 1 \mid \text{no}) \leq \frac{1}{2} - \frac{1}{\text{poly}(|x|)}$$

We have that <u>$P \subset BQP$</u> (and <u>$BPP \subset BQP$</u>)

↳ class. rand. poly. time

<u>Problems in BQP not known to be in P</u>

* Factoring
* simul. of q. systems

<u>What is a classical upper bound on BQP?</u>

(i.e., how hard it is to simulate Q. Comp. classically?)

$\Rightarrow$ Q. Comp. can be simulated in polynomial space, i.e., in PSPACE.

We can use <u>amplification</u> (i.e., many algorithm poly # of times and vote majority vote) to get this up to

$$p(\text{output} = 1 \mid \text{yes}) \geq 1 - 2^{-|x|}$$

$$p(\text{output} = 1 \mid \text{no}) \leq 2^{-|x|},$$

similarly, $p(\text{output} = 1 \mid \text{yes}) \geq \frac{1}{2} + \frac{1}{\text{poly}(|x|)}$

$$p(\text{output} = 1 \mid \text{no}) \leq \frac{1}{2} - \frac{1}{\text{poly}(|x|)}$$

We have that <u>P ⊂ BQP</u> (and <u>BPP ⊂ BQP</u>)

↳ class. rand. poly. time

<u>Problems in BQP not known to be in P</u>

* Factoring
* simul. of q. systems

<u>What is a classical upper bound on BQP?</u>

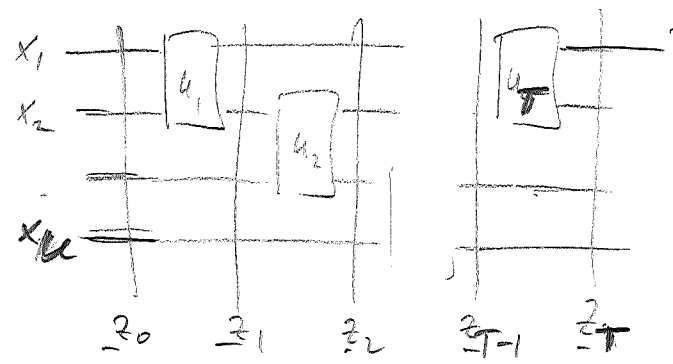
(i.e., how hard is it to simulate Q. Comp. classically?)

$\Rightarrow$ Q. Comp. can be simulated in polynomial space,

i.e., in PSPACE

BQP ⊂ PSPACE:

Proof via path integral:



$P_{yes} = ?$

$(T = poly(n))$

$$P_{yes} = \sum_{z_T^1, \ldots, z_T^n} |\langle 1 z_T^2 \ldots z_T^n | \psi \rangle|, \text{ with}$$

$$\langle z_T | \psi \rangle = \langle z_T | U_T \cdot \ldots \cdot U_1 | x_1 \ldots x_n \rangle$$

$$= \sum_{z_1, \ldots, z_{T-1}} \underbrace{\langle z_T | U_T | z_{T-1} \rangle \langle z_{T-1} | U_{T-1} | z_{T-2} \rangle \cdots \langle z_0 | U_1 | x_1 \ldots x_n \rangle}_{\circledast}$$

* $\langle z_t | U_t | z_{t-1} \rangle$ can be computed efficiently

(U_t only acts on few qubits)

* Product $\circledast$ can be computed efficiently

$*$ $\sum_{z_1,\dots,z_{T-1}}$: runs over $n \times (T-1)$ bits, i.e., $2^{n\times(T-1)}$ strings.

$\Rightarrow$ can be solved iteratively ("for-loop") –

– exponential time, but only space $\sim n \times (T-1)$ needed to store value of addend

$\Rightarrow$ P_{yes} can be computed w/ $\mathrm{poly}(n \cdot T) = \mathrm{poly}(n)$ time!

Note: To get any $\#k = \mathrm{poly}(n)$ of digits for P_{yes}, we need at most $k + (n \times T)$ digits for each $\circledast$, and thus for each number

$\Rightarrow$ $\mathrm{poly}(n)$ memory also for numbers.

$\Rightarrow$ <u>BQP $\subset$ PSPACE</u>!

(Note: This is likely not a good Journal – the same idea would also work for non-unitary circuits, post-selection, etc.)

(A slightly tighter bound is in fact BQPCPP)

Can we identify hard problems for quantum computers –
i.e. a quantum version of NP?

NP: (Potentially) hard to solve, but with class. proof which can be checked efficiently by class. computer

Quantum NP: Pot. hard, but there

exists a { classical / quantum } proof which can be checked eff. by a quantum computer.

q. proof: "QMA"

cl. proof: "QCMA"

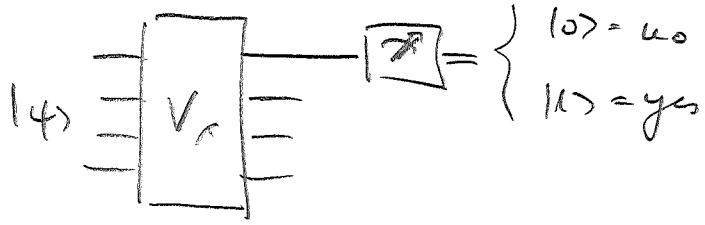
(Note: MA – "Merlin Arthur" is as NP, but with a probabilistic verifier)

QMA ("Quantum Merlin Arthur")

Class of problems where for yes instances there exists a "quantum proof" $|\psi\rangle$ which can be eff. checked by a quantum computer.

More formally:

$L \in$ QMA iff there exists a (uniform!) family of quantum circuits $V_x(|\psi\rangle)$, of size poly$(|x|)$



s.t.: $x \in L \Rightarrow \exists |\psi\rangle$: prob$(V_x(|\psi\rangle) = \text{yes}) \geq 2/3$

$x \notin L \Rightarrow \forall |\psi\rangle$: prob$(V_x(|\psi\rangle) = \text{yes}) \leq 1/3$.

(Again, any prob. with $1/\text{poly}$ separation is ok.)

What is a typical QMA problem?

The "k-local Hamiltonian" problem:

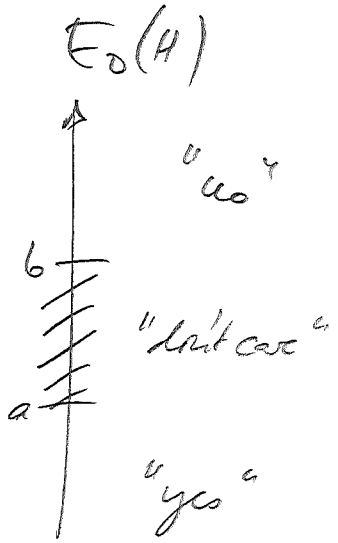
Given:
- a number n
- a k-local Hamiltonian

$$H = \sum_i h_i,$$

i.e. each $h_i = h_i^\dagger$ acts on at most k qubits (i.e. $h_i \equiv (h_i)_k \otimes \mathbb{1}_{rest}$).

- Thresholds a, b with $b - a \geq \frac{1}{poly(n)}$

Let $E_0(H)$ be the ground state energy of H, i.e. the smallest eigenvalue of H.



Question: Is $E_0(H) \leq a$ ("yes")

or $E_0(H) \geq b$ ("no")

given the promise that $E_0(H) \notin (a; b)$

(Colloquially: Compute ground state energy of H up to $\frac{1}{poly(n)}$ precision.)

Why is "k-local Hamiltonian" in QMA?

Given a state $|\psi\rangle$, we can estimate $\langle\psi|H|\psi\rangle$ with a quantum computer:

- using phase estimation for $U = e^{2\pi i H t}$ or
- by rand. selecting i and meas. $\langle\psi|h_i|\psi\rangle$ (e.g. by a proj. meas. in the eigenbasis)

$\Rightarrow$ allows to distinguish $\langle\psi|H|\psi\rangle \le a$ vs. $\langle\psi|H|\psi\rangle \ge b$ with at least $\frac{1}{\text{poly}}$ prob. $\Rightarrow$ can be amplified.

Yes instance: Proof $\equiv$ Ground state $|\psi_0\rangle$, $\langle\psi_0|H|\psi_0\rangle \le a$.

No instance: $\forall|\psi\rangle$: $\langle\psi|H|\psi\rangle \ge b$.

$\Rightarrow$ k-local Hamiltonian is in QMA.

But: k-local Ham. is also QMA-complete, i.e., among the hardest problems in QMA!

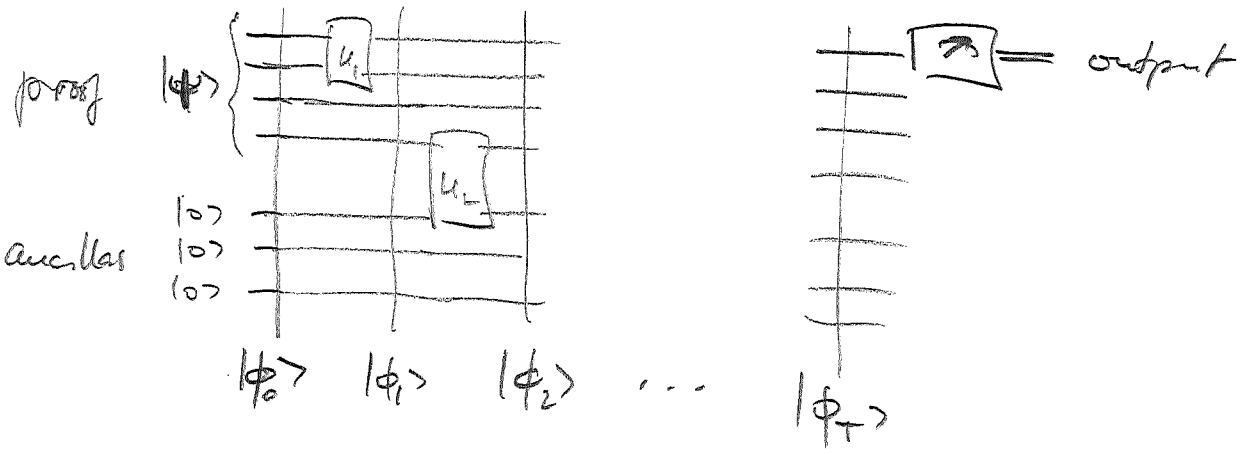
$\Rightarrow$ Finding ground states is (probably) hard even for quantum computers!

(Note that time evolution can be simulated in BQP!)

How to prove QMA-completeness of k-loc. Ham?

Follow ideas from Cook-Levin proof:

General QMA problem ⟷ verifier circuit.



Build again "history state".

Two options: ~~$|\phi_0\rangle \otimes |\phi_1\rangle \otimes \dots \otimes |\phi_T\rangle$~~ ← not good-Gate!

or $|\phi_0\rangle \oplus |\phi_1\rangle \oplus \dots \oplus |\phi_T\rangle$

$$\equiv |\Phi\rangle = \sum |\phi_t\rangle |t\rangle$$

Build Hamiltonian to ensure (1 in ground state)

(i) correct initialization

(ii) correct time evolution

(iii) output = yes.

(i) correct initialization:

$$H_{init} = \sum_i h_i \otimes |0\rangle\langle 0|_t \quad ; \quad h_i = |1\rangle\langle 1| \text{ on ancillas}$$

$\Rightarrow$ penalize $|1\rangle$ - ancillas

(ii) propagation:

$$H_{prop} = \sum_t -\left(U_t |t\rangle\langle t-1| + h.c.\right) + |t\rangle\langle t| + |t-1\rangle\langle t-1|$$

$$\equiv \begin{pmatrix} \mathbb{1} & -U_t^\dagger \\ -U_t & \mathbb{1} \end{pmatrix}$$

(t, t-1)

$\Rightarrow$ penalize "incorrect propagation" - ground space spanned by $|\phi_t\rangle = U_t |\phi_{t-1}\rangle$.

$$H_{final} = |0\rangle\langle 0|_L \otimes |1\rangle\langle 1|_t$$

$\Rightarrow$ penalize "no" output.

$$\boxed{H_{QMA} = H_{init} + H_{prop} + H_{final}}$$

H_{QTCA} has energy ≈ 0 if there ex. a $|\psi\rangle$ which is accepted w. high prob.

Otherwise, $E_0(H_{QTCA}) \geq 1/\text{poly}(n)$

$\Rightarrow$ H_{QTCA} QTCA-complete!

(Note: The real proof is quite involved!)

Notes:

* time-register has $\log n$ qubits
  $\Rightarrow$ $\log n$-local Ham.
  $\Rightarrow$ can be made 5-local by using a unary encoding of $|t\rangle$.

* simpler Hamiltonians can be constructed:
  - 2D lattice of qubits w/ NN interaction
  - 1D chains (!)

* Why can't we use a history state
  $|\phi_0\rangle \otimes |\phi_1\rangle \otimes \ldots \otimes |\phi_T\rangle$?

→ impossible to ensure

$$|\phi_t\rangle = U_t |\phi_{t+}\rangle .$$

E.g. for 1 qubit, $U_t = Y$:

- $|\phi\rangle \propto |\phi\rangle$ must be ground state for all $|\phi\rangle$.
- $|\phi\rangle \propto |\phi'\rangle$ spans sym. subspace

⟹ $|0\rangle|1\rangle + |1\rangle|0\rangle$ is also a ground state ✓.