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

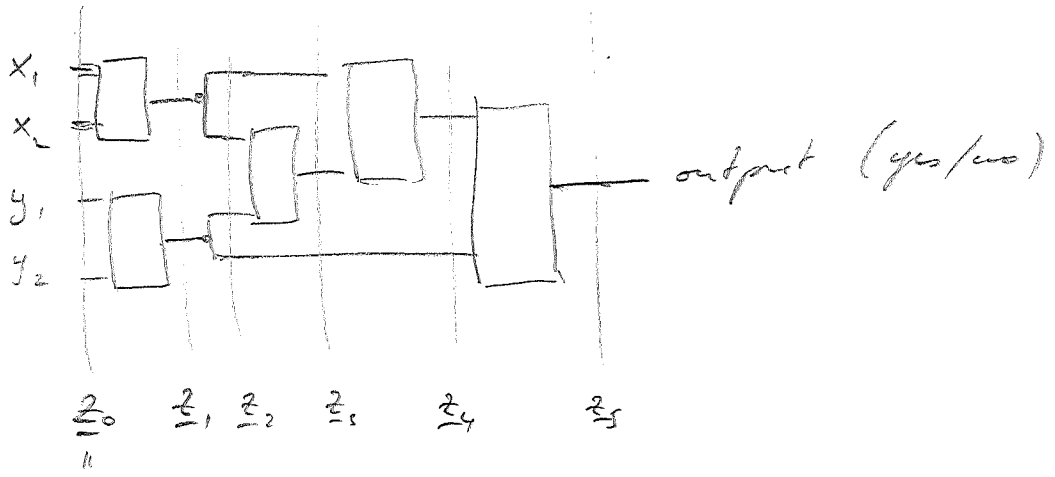
The Cook-Levin-theorem: 3-SAT is NP-complete

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.