Lecture 13: Complexity theory & many-body problems

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

Difficulty is typ. measured in terms of resources needed (memory, time) as a function of the problem size (= #bits needed to specify the problem).

For simplicity, we typ. focuses on decision problems (yes/no answers),
$\leftrightarrow$ problem corresponds to $f(\{0,1\}^{\otimes N}) \longrightarrow \{0,1\}$.

(Note: Not a severe restriction — many problems have an decision version w/ ess. same difficulty.)

Note: CS terminology:

$L = \{x \mid f(x) = 1\}$ is called "Language"

There are many different models for computers: circuits (= gates), processors ("real" computers), Turing machines (tape + head), ...

All known "reasonable" models have the same power $\leftrightarrow$

"Church-Turing thesis"

Classical complexity classes:

P ("polynomial time"): $f(x)$ can be computed in a time $\text{poly}(|x|)$ (↳ length of x)

(CS terminology: LEP iff it can be decided in poly time.)
These are the "efficiently solvable problems."

Examples in P:

* multiplication, addition, ... (or dec. version)
* primality testing
* ground state of 1D classical Hamiltonians (→ homework)

NP ("non-deterministic polynomial time"):

Problem might be hard to solve, but there is an efficiently checkable proof y: There exists function $g(x,y) \longmapsto \{0,1\}$ (↑ poly(|x|) bits)

$x \in L$: $\exists y$ s.t. $g(x,y)=1$ ("proof accepted")

$x \notin L$: $\forall y$ $g(x,y)=0$ ("no proof exists")

Examples:

* Graph coloring (color vertices w/ no two vertices w/ same color):

"yes" ≡ coloring exists → proof = a valid coloring (can be checked efficiently)

"no" ≡ no coloring exists

* k-SAT: variables $x_1, \dots, x_K$; "clauses" $g_j = x_{a_j} \vee x_{b_j} \vee x_{c_j}$.

Q: Does there exist $x_1, \dots, x_K$ s.th. $g_j = 1 \; \forall j$?

Proof in "yes" case: Satisfying assignment $x_1, \dots, x_K$.

* Prime factor decomposition
* Graph isomorphism: are two graphs isomorphic?

Note: • NP problems can be hard (P≠NP - conjecture!), but not arbitrarily hard - there are problems w/out proofs, e.g. games ("who wins a chess")

• P ⊂ NP: Any P problem is also a NP!

How can we identify the "hardest" problems in NP (which are hard unless P=NP)?

→ NP-completeness: A problem is NP-complete if we can map ("reduce") any problem in NP to it. (i.e.: If we could solve the NP-complete problem, we could solve any NP problem.

Examples for complete problems:

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

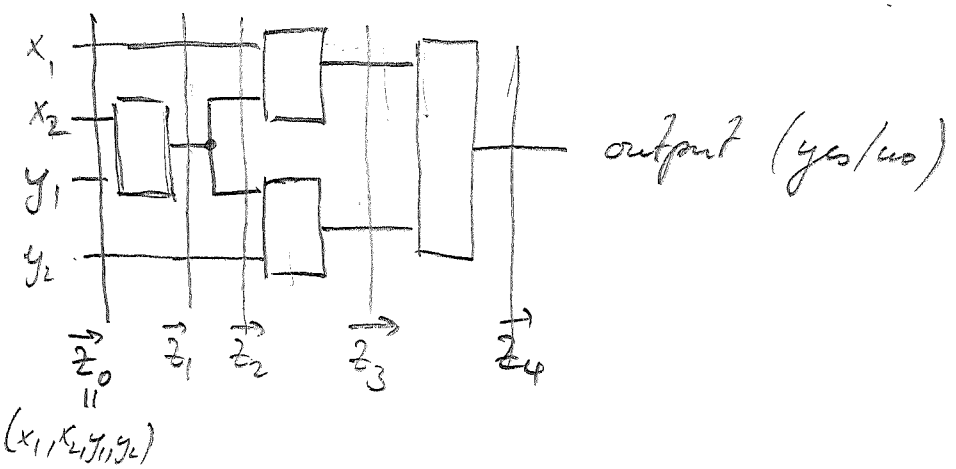
Non-complete problems:

* factoring, graph isomorphism

How can we show that 3-SAT is NP-complete? (Cook & Levin)

General NP problem $\longleftrightarrow$ verifier $g(x,y)$

↑ circuit.



"history state" $(\vec{z}_0, \vec{z}_1, \dots, \vec{z}_4)$ for verifier checking proof y.

Now construct clauses as follows:

* For each input x_i, add clause which ensures the right value x_i in $\vec{z}_0$.
* For each gate, add clauses which ensures that the gate is performed correctly (acting on $\vec{z}_0$ & $\vec{z}_1$, $\vec{z}_1$ & $\vec{z}_2$, etc.)
* Add a clause which requires the output ($\vec{z}_4$) to be "yes".

$\Rightarrow$ This gives a 3-SAT problem.

3-SAT satisfyable $\Longleftrightarrow$ $\exists y : g(x,y) = 1$

i.e.: this 3-SAT is equivalent to the original problem

$\Longrightarrow$ 3-SAT NP-complete!

Implications for Hamiltonians:

* map clauses to Ham. terms ($h=0$ if clause satisfied, else $h=1$)
* 3-body Ham. in 2D geometry NP-complete!
* using different techniques: 2-body Ham. (here w/ magnetic field) still NP-complete.
* But: 1D-Ham.: easy (P) ($\rightarrow$ Homework)

Remark: Complexity results do not mean we can't solve certain systems in practice — first, in practice we face special instances of the problem with special properties, and second, complexity results are worst-case results — average instances might still be easy.

However, these results tell us that we cannot find provably working algorithms for many-body systems without additional assumptions!