## What does it mean to be "the hardest problem in NP"?

DEFINITION (NP-HARD and NP-COMPLETE)
The language $L \subseteq \Sigma^*$ is $\boxed{\text{NP-HARD}}$ if for every $L' \in NP$ it holds that $L' \leq_p L$ (i.e., there is a polynomial-time-computable function $g : \Sigma^* \to \Sigma^*$ such that

$$x \in L' \iff g(x) \in L$$

$L$ is $\boxed{\text{NP-COMPLETE}}$ if in addition $L \in NP$

$L$ is as hard as any problem in NP, since any efficient algorithm for $L$ can be used to decide any language in NP efficiently

LEMMA

1. If $L_1 \leq_p L_2$ and $L_2 \leq L_3$, then $L_1 \leq_p L_3$ (TRANSITIVITY)
2. If $L$ is NP-hard and $L \in P$, then $P = NP$
3. If $L$ is NP-complete, then $L \in P$ iff $P = NP$

Proof Exercise, or see textbook.

But do NP-complete problems exist? Not clear from the definition. But indeed NP-complete problems are all over the place in computer science, mathematics, physics, chemistry, biology, economics, industry ...

The most important ones (at least historically) are variants of $\boxed{\text{SATISFIABILITY}}$ (or SAT for short)

CNF formula (conjunctive normal form)

$$(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee \neg x_3)$$
$$\wedge (\neg x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3)$$

Variables $x_i$ (or $x, y, z$) set to true $= 1$ or false $= 0$

Logical connectives AND $\wedge$, OR $\vee$, NOT $\neg$

(or sometimes write $\bar{x}$ for $\neg x$)

(Disjunctive) clause $C = x_1 \vee \neg x_2 \vee \neg x_3$

satisfied if one literal assigned to true

CNF formula conjunction of clauses

$$F = C_1 \wedge C_2 \wedge \cdots \wedge C_m = \bigwedge_{i=1}^{m} C_i$$

Satisfied if all clauses $C_i$ are satisfied

$$\boxed{SAT = \{ F \mid F \text{ is a satisfiable CNF formula} \}}$$

$k$-CNF formula: Each clause has $\leq k$ literals

$$\boxed{k\text{-}SAT = \{ F \mid F \text{ is a satisfiable } k\text{-CNF formula} \}}$$

COOK-LEVIN THEOREM (1971 and 1973, respectively)

1. SAT is an NP-complete problem
2. 3-SAT is an NP-complete problem

Focus on 1 — 2 is easy corollary.

Clearly, SAT $\in$ NP. A satisfying assignment to the variables is a short witness that is easy to verify

Need to show: For any $L \in NP$, exists efficient reduction $g$ such that

$$x \in L \iff g(x) \text{ is a satisfiable CNF formula}$$

What can we do to prove this?

- Only thing we know is that exists Turing machine $M_L(x, y)$ such that

  $x \in L \iff$ Exists $y$ of length $\leq p(|x|)$ such that $M_L(x, y) = 1$

- Write computation of such TM $M_L$ on $x$ as a CNF formula

- Show that if $x \in L$, then can plug in witness $y$ such that formula describing TM computation is satisfied

Think of alphabet $\Sigma$ as $\{0, 1\}$ (can always re-encode symbols)

Think of input $x$ as given.

Define Boolean function $f_x$ by

$$\boxed{f_x^L(y) = \begin{cases} 1 & \text{if } M_L(x, y) = 1 \\ 0 & \text{otherwise} \end{cases}}$$

## PROPOSITION 4

Any Boolean function $f : \{0,1\}^\ell \to \{0,1\}$ can be expressed as CNF formula of size $\leq \ell \cdot 2^\ell$

(size := total # literals in formula counted with repetitions)

## Proof sketch

Consider assignment $\alpha$ s.t. $f(\alpha) = 0$

Write down clause $C_\alpha$ falsified <u>exactly</u> by this one assignment

Let the CNF formula be $F = \bigwedge_{\alpha \in f^{-1}(0)} C_\alpha$

**EXAMPLE** PARITY $(x_1, x_2, x_3)$ = odd # variables true

Truth table

| $x_1$ | $x_2$ | $x_3$ | PARITY | Clauses |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | $(x_1 \vee x_2 \vee x_3)$ |
| 0 | 0 | 1 | 1 | |
| 0 | 1 | 0 | 1 | |
| 0 | 1 | 1 | 0 | $\wedge\ (x_1 \vee \neg x_2 \vee \neg x_3)$ |
| 1 | 0 | 0 | 1 | |
| 1 | 0 | 1 | 0 | $\wedge\ (\neg x_1 \vee x_2 \vee \neg x_3)$ |
| 1 | 1 | 0 | 0 | $\wedge\ (\neg x_1 \vee \neg x_2 \vee x_3)$ |
| 1 | 1 | 1 | 1 | |

This CNF formula evaluates to true precisely when # true variables odd

Proof attempt 1 for Cook-Levin Theorem:

Given $L \in NP$, verifier $M_L(x,y)$, and input $x$

Consider $f_x^L(y)$

Use Proposition 4 to generate CNF formula $F_x$

$F_x$ is satisfiable $\Longleftrightarrow$ Exists $y$ s.t. $M_L(x,y) = 1$
This is our reduction $g$! QED

OR IS IT? What is the size of $F_x$?
$p(|x|) \cdot 2^{p(|x|)}$ — exponential!
So $g$ will run in exponential time — too slow!

Use that Turing machine computations are LOCAL
Only depend on current state and currently read symbols

Simplifying assumptions (but justified):

1. For any $L \in NP$, fix polynomial $p_L$ such that witnesses should have length exactly $p_L(|x|)$
2. Turing machines have two tapes, input and work/output
3. Turing machine is $\boxed{\text{OBLIVIOUS}}$: head movements do not depend on tape contents, only on input length (which is $|x| + |y|$)

Can run $M_L$ on $x$ and $0^{p(|x|)} = \overbrace{000\ldots0}^{p(|x|)}$
to determine head positions at every time step
and build table

~~At most~~ quadratic loss in running time

(See Arora-Barak Ch 1 for details)

## Notation

$Q$ : Turing machine states (= "lines in program")

$\Sigma$ : Alphabet (containing $0, 1, \sqcup$ = blank etc)

$u$ : Input $x$ and $y$ concatenated

All symbols in $\Sigma$ can be encoded in binary

If $|\Sigma| = s$, need $\lceil \log_2 s \rceil$ bits

SNAPSHOT $z = \langle a, b, q \rangle \in \Sigma \times \Sigma \times Q$

captures what determines Turing machine action at given time step

$a$ : symbol read on input tape

$b$ : symbol read on work/output tape

$q$ : current TM state

Since $Q$ is also finite, snapshot $z$ can be encoded as binary string of fixed length, say $c$ bits [choose $c \geq \lceil \log_2 s \rceil$]

Snapshot $z_i$ at time $i$ depends on:

(a) state at time $i-1$

(b) contents at current locations of tapes

Suppose we're given sequence of snapshots $z_1, z_2, z_3, \ldots, z_t$ claimed to describe computation by $M_L$. How to verify?

INSIGHT: We can verify each $z_i$ by only local checks

To check $z_i = \langle a_i, b_i, q_i \rangle$, only need to look at

1. $z_{i-1} = \langle a_{i-1}, b_{i-1}, q_{i-1} \rangle$ — tells us if jump to state $q_i$ correct

2. $u_{inputpos(i)}$ — The input tape head is at position $inputpos(i)$ at time $i$ (which we have computed in a table), so check $u_{inputpos(i)} = a_i$

3. $z_{prev(i)}$ — Time $prev(i)$ is the last time the work tape head was at its current position (which we have also computed in a table), so check that symbol <u>written</u> to output tape at time $prev(i)$ as specified by $z_{prev(i)} = \langle a_{prev(i)}, b_{prev(i)}, q_{prev(i)} \rangle$ is the same as $b_i$.

$\boxed{\text{(1) – (3) \underline{uniquely} determine } z_i}$

So there is a Boolean function

$$step_i(z_i, z_{i-1}, u_{inputpos(i)}, z_{prev(i)})$$

that evaluates to true precisely when transition at time step $i$ correct

Function of $\leq 4c$ bits = constant

Apply Proposition 4 $\Rightarrow$ constant-size formula!

Running time of $M_L(x,y)$ is <u>exactly</u> $q(|x|+|y|)$ for some polynomial $q$, which is $q^*(|x|)$ for some other polynomial

$$q^*(|x|) = q(|x| + p_L|x|)$$

Let our <u>reduction</u> write down CNF formula $F_x$ as follows

1. <u>subformula INPUT(x)</u> saying that first $|x|$ symbols on input tape must match $x$.

2. <u>subformula START ($z_1$)</u> encoding that the starting position of $M_L$ is correct

3. <u>subformulas STEP(i)</u> for $i = 2, 3, \ldots, q^*(|x|)$ saying that snapshot $z_i$ is correct given $z_{i-1}$, $u_{\mathrm{inputpos}(i)}$, and $z_{\mathrm{prev}(i)}$ (where we can look up inputpos(i) and prev(i) in tables)

4. <u>subformula ACCEPT</u> saying that the final state is that of an accepting computation of $M_L$ (e.g., 1 is written in first position of work tape and all other positions blank)

$$\boxed{F_x = \mathrm{INPUT}(x) \wedge \mathrm{START}(z_1) \wedge \bigwedge_{i=2}^{q^*(|x|)} \mathrm{STEP}(i) \wedge \mathrm{ACCEPT}}$$

Subformulas 1, 2, 4 easy — just fixing bits to values

"$v = w$" $\quad (\neg v \vee w) \wedge (v \vee \neg w)$

Subformula 3

size $\quad q^*(|x|) \cdot \overbrace{(\text{exponential in } c)}^{\text{constant}}$

Can be computed in polynomial time

- First run $M_L(x, 0^{p_L(|x|)})$ to compute tables "inputpos" and "prev"
- Then output CNF formula $F_x$, where subformula 3 is what takes time

$F_x$ is satisfiable if and only if exists $y$ such that $M_L(x,y) = 1$, i.e., if and only if $x \in L$ QED (for real) $\blacksquare$

Reducing from SAT to 3-SAT is straightforward exercise (or see textbook)

<u>Two observations</u>

1. If $M_L$ runs in time $T(|x|)$, formula $F_x$ can be made very small — $O(T \log T)$
2. From satisfying assignment to $F_x$, can read off witness $y$ for $x$

This is called a LEVIN REDUCTION

To prove that a language $L$ is NP-complete, we need to do two things

(i) Show $L \in NP$ (usually easy)

(ii) Reduce from SAT or 3-SAT (or from some other already known NP-complete problem) to $L$

We will now see some such reductions. But first one more definition

DEFINITION 5: COMPLEMENT CLASS

For a language $L \subseteq \Sigma^*$, the COMPLEMENT of $L$ is $\overline{L} = \Sigma^* \setminus L$

DEFINITION 6: coNP

$$\boxed{coNP = \{ L \mid \overline{L} \in NP \}}$$

Aside: If strings in $L$ encode objects such as formulas or graphs, then we usually think of $\overline{L}$ as only containing correctly encoded instances.

That is, $L$ and $\overline{L}$ will both be sets of formulas or graphs satisfying or not satisfying some property, respectively, while "syntax error" strings are not contained in either $L$ or $\overline{L}$.

This is just a technical convention that doesn't really make much

Note that coNP is <u>not</u> the complement of NP — the intersection is non-empty! E.g.,

$$P \subseteq NP \cap coNP \quad \text{(why?)}$$

<u>EXAMPLE</u> $\boxed{\text{TAUTOLOGY}}$ (example ⑦ above) is in coNP. If F is <u>not</u> a tautology, then this is witnessed by an assignment falsifying the formula

$\boxed{\text{UNSAT}}$ = { F | F is an unsatisfiable CNF formula } is also in coNP

In fact, both of these languages are coNP-complete — any other language $L \in coNP$ can be efficiently reduced to them

<u>Proof sketch</u> for UNSAT: Given $L \in coNP$, run Cook-Levin reduction on $\overline{L}$

$$x \in L \iff x \notin \overline{L} \iff F_x \notin SAT$$
$$\iff F_x \in UNSAT$$

<u>How is coNP related to NP?</u>
Could there be short certificates for UNSAT that somehow "compress information about exponentially many assignments"?
<u>Most researchers believe</u> $\boxed{NP \neq coNP}$
but this is a wide-open question!