P vs. NP

Xin Li

lixints@cs.jhu.edu
Topics

- Equivalence of two definitions of NP.

- A new kind of reduction.

- NP-completeness and Cook-Levin Theorem.
Computation of a non-deterministic TM

- Whenever the transition function has 2 possible outcomes
  - The TM splits into 2 copies/branches, with each copy taking one outcome.

- The TM accepts an input if $\exists$ a copy that accepts the input.

- The TM rejects an input if all copies reject the input.
Running time of a non-deterministic TM

• Similarly, an NDTM may not halt on an input for some branch of computation.

• We prefer NDTMs that halt on all branches of computation (decider).

• For a function $T: N \rightarrow N$, say an NDTM $N$ runs in time $T(n)$ if for any input of length $n$, all branches of $N$ halt within $T(n)$ steps.
NTIME and alternative definition of NP

• Let $T: \mathbb{N} \rightarrow \mathbb{N}$ be some function. Let NTIME($T(n)$) be the set of all languages that can be decided by some NDTM in time $O(T(n))$.

• Let $NP = \bigcup_{c \geq 0} NTIME(n^c)$, i.e., all languages decidable by some NDTM in polynomial time of the input length.
Equivalence of two definitions of NP

• Recall:

• A language $L$ is in the class $NP_1$ if

  $\exists$ a polynomial function $P: N \rightarrow N$ and a polynomial time TM $M$ s.t. $\forall x \in \{0, 1\}^*$

  $x \in L$ iff $\exists u \in \{0, 1\}^{P(|x|)}$ s.t. $M(x, u)=1$ (accept)

• $NP_2=\bigcup_{c \geq 0} NTIME(n^c)$

• Show $NP_1 = NP_2$, i.e., $NP_1 \subseteq NP_2$ and $NP_2 \subseteq NP_1$. 
Proof part 1

• $NP_1 \subseteq NP_2$, i.e., $\forall$ language $L$, $L \in NP_1 \Rightarrow L \in NP_2$.

  $\exists$ a polynomial function $P$: $N \rightarrow N$ and a polynomial time TM $M$ s.t. $\forall x \in \{0, 1\}^*$
  $x \in L$ iff $\exists u \in \{0, 1\}^{P(|x|)}$ s.t. $M(x, u)=1$ (accept)

• Goal: build an NDTM $N$ to decide $L$ based on the above definition.

• Idea: $N$ can non deterministically guess the certificate $u$ and then run $M(x,u)$. 
Proof part 2

• $NP_2 \subseteq NP_1$, i.e., $\forall$ language $L$, $L \in NP_2 \Rightarrow L \in NP_1$.

  $\exists$ a polynomial function $P: \mathbb{N} \rightarrow \mathbb{N}$ and a polynomial time TM $M$ s.t. $\forall x \in \{0, 1\}^*$
  $x \in L$ iff $\exists u \in \{0, 1\}^{P(|x|)}$ s.t. $M(x, u)=1$ (accept)

• Goal: If $L$ can be decided by an NDTM in time $\text{poly}(n)$, show that $x \in L$ iff $\exists$ a certificate.

• Idea: The certificate $u$ is the non-deterministic choices of the transition function that lead to acceptance (the length is $\text{poly}(n)$).
From Previous Lecture

• $P \subseteq NP \subseteq EXP$

• At least one is proper.

• $P$ vs. $NP$ is one of the unsolved millennium problems
Ideas to prove $P$ vs. $NP$

• To prove $P=NP$: show all languages in $NP$ can be decided in polynomial time.

• To prove $P \neq NP$: show one language in $NP$ that cannot be decided in polynomial time.

• Unify: for both cases only need to study one specific language, i.e., the hardest language in $NP$. 
The Hardest language in NP

• How to define hardest?

• Use reduction: if $A \leq_m B$ then $B$ is harder than $A$.

• A language is hardest if all other languages in this class can be reduced to it.

• Need to use a modified reduction related to the classes.
The Reduction

- When studying two complexity classes, the reduction should not be more powerful than the weaker class.

- $P$ vs. $NP$: polynomial time reduction.
Mapping Reduction

- Say a language $A$ is mapping reducible to a language $B$, written $A \leq_m B$, if

  $$\exists \text{ a computable function } f: \Sigma^* \rightarrow \Sigma^* \text{ s.t. } \forall w, w \in A \text{ iff } f(w) \in B$$

if and only if

never cross the boundary
Polynomial Time Reduction

• Say a language A is polynomial time (Karp) reducible to a language B, written $A \leq_p B$, if

$$\exists \text{ a polynomial time computable function } f: \Sigma^* \rightarrow \Sigma^* \text{ s.t. } \forall w, w \in A \iff f(w) \in B$$

if and only if

never cross the boundary
Properties of poly-time reduction

• If $A \leq_p B$ then

• If $B \in P$, then $A \in P$.

• If $A \leq_p B$ and $B \leq_p C$, then $A \leq_p C$. 
NP-hard and NP-complete

• A language $L$ is NP-hard if $\forall A \in NP, A \leq_p L$ (L is harder than all languages in NP).

• A language $L$ is NP-complete if $L$ is NP-hard and $L \in NP$ (L is the hardest language in NP).

• If $L$ is NP-hard and $L \in P$, then $P=NP$.

• If $L$ is NP-complete, then $L \in P$ iff $P=NP$. 
Implications

• If $L$ is NP-complete, then $L \in P$ iff $P=NP$.

• The P vs. NP question boils down to study one language (an NP-complete language).

• But, does such a language exist?
Cook-Levin Theorem (1971)

- The first NP-complete language.

- Soon after, Karp showed many other NP-complete languages.

- Boolean formulas and SAT.
Cook-Levin Theorem (1971)

• Define SAT=\{all satisfiable CNF formulas\}.

• Define 3SAT=\{all satisfiable 3CNF formulas\}.

• Cook-Levin Theorem: SAT is NP-complete, 3SAT is NP-complete.
Proof of Cook-Levin Theorem

• SAT is NP-complete

• Need to show:

  (1) SAT ∈ NP

  (2) ∀ L ∈ NP, L ≤_p SAT

• SAT ∈ NP:

  Certificate: a specific satisfying assignment.
SAT is NP-complete

- $\forall L \in NP, \ L \leq_p SAT$

- Can only use the definition of NP.
SAT is NP-complete

• A language $L$ is in the class NP if

  $\exists$ a polynomial function $P: \mathbb{N} \rightarrow \mathbb{N}$ and a polynomial time TM $M$ s.t. $\forall x \in \{0, 1\}^*$
  $x \in L \iff \exists u \in \{0, 1\}^{P(|x|)}$ s.t. $M(x, u)=1$

• $\forall L \in NP, \ L \leq_p SAT$ if

  $\exists$ a polynomial time computable function $f: \{0, 1\}^* \rightarrow \{0, 1\}^*$ s.t. $\forall x \in \{0, 1\}^*$
  $x \in L \iff f(x)=\psi_x \in SAT.$

• Only need to show $\psi_x \in SAT \iff \exists u \in \{0, 1\}^{P(|x|)}$ s.t. $M(x, u)=1.$
SAT is NP-complete

• Only need to show $\psi_x \in \text{SAT} \iff \exists u \in \{0, 1\}^{P(|x|)} \text{ s.t. } M(x, u) = 1$.

• Idea: express the computation of $M(x, u)$ as a CNF formula $\psi_x$ with polynomial size.

• Important idea: computation is local.
The reduction

• $3SAT \in NP$

• Show $\text{SAT} \leq_p 3\text{SAT}$

• Given any CNF formula $\psi$, construct in poly time another 3CNF $\psi'$ s.t. $\psi \in \text{SAT}$ iff $\psi' \in 3\text{SAT}$. 
3SAT is NP-complete

- If $\psi = C_1 \land C_2 \land \ldots \land C_m$ is already a 3CNF, done.

- Otherwise WLOG assume $C_1 = A_1 \lor B_1$ has more than 3 literals, where $B_1$ has 2 literals.

- Introduce a new variable $u$ and two clauses $D_1 = A_1 \lor u$ and $D_2 = B_1 \lor \neg u$

- Let $\psi' = D_1 \land D_2 \land C_2 \land \ldots \land C_m$
3SAT is NP-complete

• Claim: \( \psi \) is satisfiable iff \( \psi' \) is satisfiable.
  \[
  C_1 = A_1 \lor B_1, \quad D_1 = A_1 \lor u, \quad D_2 = B_1 \lor \neg u
  \]

• \( \psi \) is satisfiable \( \implies \) \( \psi' \) is satisfiable

• An assignment \( z \) satisfies \( \psi \), then \( C_1 = A_1 \lor B_1 = 1 \) under \( z \).

• Can assign a value to \( u \) based on \( A_1, B_1 \) so that both \( D_1 \) and \( D_2 \) are satisfied.
3SAT is NP-complete

• $\psi'$ is satisfiable $\Rightarrow$ $\psi$ is satisfiable

$$ C_1 = A_1 \lor B_1, \ D_1= A_1 \lor u , \ D_2= B_1 \lor \neg u $$

• An assignment $z$ satisfies $\psi'$, then $D_1 = 1$ and $D_2 = 1$ under $z$.

• One of $A_1, B_1$ must be 1 under $z$ $\Rightarrow$ $C_1 = A_1 \lor B_1 = 1$ under $z$. 
3SAT is NP-complete

• $C_1 = A_1 \lor B_1$, $D_1 = A_1 \lor u$, $D_2 = B_1 \lor \neg u$

• $D_2$ has 3 literals, $D_1$ has one less literal than $C_1$.

• Can keep doing this until $C_1$ turns into a sequence of clauses with 3 literals

• Do the same for any other clause, the reduction is in poly time.