CSE 421

Polynomial Time Reductions, NP, NP-Completeness

Shayan Oveis Gharan
\( \leq_p^1 \) Reductions

Here, we will always use a restricted form of polynomial-time reduction often called Karp or many-to-one reduction

\( A \leq_p^1 B \): if and only if there is an algorithm for A given a black box solving B that on input \( x \)

- Runs for polynomial time computing an input \( f(x) \) of B
- Makes one call to the black box for B for input \( f(x) \)
- Returns the answer that the black box gave

We say that the function \( f(.) \) is the reduction
Example 1: Indep Set \leq_p Clique

**Indep Set**: Given \(G=(V,E)\) and an integer \(k\), is there \(S \subseteq V\) s.t. \(|S| \geq k\) and no two vertices in \(S\) are joined by an edge?

**Clique**: Given a graph \(G=(V,E)\) and an integer \(k\), is there \(S \subseteq V\), \(|U| \geq k\) s.t., every pair of vertices in \(S\) is joined by an edge?

**Claim**: Indep Set \leq_p Clique

**Pf**: Given \(G = (V, E)\) and instance of indep Set. Construct a new graph \(G' = (V, E')\) where \(\{u, v\} \in E'\) if and only if \(\{u, v\} \notin E\).

\[1\] \[2\] \[3\] \[4\] \[5\]

S is an independ set in G

\[1\] \[2\] \[3\] \[4\] \[5\]

S is an Clique in G'
Example 2: Vertex Cover $\leq_p$ Indep Set

**Vertex Cover**: Given a graph $G = (V, E)$ and an integer $k$, is there a vertex cover of size at most $k$?

**Claim**: For any graph $G = (V, E)$, $S$ is an independent set iff $V - S$ is a vertex cover

**Pf:**

$\Rightarrow$ Let $S$ be an independent set of $G$

Then, $S$ has at most one endpoint of every edge of $G$

So, $V - S$ has at least one endpoint of every edge of $G$

So, $V - S$ is a vertex cover.

$\Leftarrow$ Suppose $V - S$ is a vertex cover

Then, there is no edge between vertices of $S$ (otherwise, $V - S$ is not a vertex cover)

So, $S$ is an independent set.
Example 3: Vertex Cover $\leq_p$ Set Cover

**Set Cover**: Given a set $U$, collection of subsets $S_1, \ldots, S_m$ of $U$ and an integer $k$, is there a collection of $k$ sets that contain all elements of $U$?

**Claim**: Vertex Cover $\leq_p$ Set Cover

**Pf**: Given $(G = (V, E), k)$ of vertex cover we construct a set cover input $f(G, k)$

- $U = E$
- For each $v \in V$ we create a set $S_v$ of all edges connected to $v$

This clearly is a polynomial-time reduction

So, we need to prove it gives the right answer
Example 3: Vertex Cover $\leq_p$ Set Cover

Claim: Vertex Cover $\leq_p$ Set Cover

Pf: Given $(G = (V, E), k)$ of vertex cover we construct a set cover input $f(G, k)$

- $U = E$
- For each $v \in V$ we create a set $S_v$ of all edges connected to $v$

Vertex-Cover $(G, k)$ is yes $\Rightarrow$ Set-Cover $f(G, k)$ is yes

If a set $W \subseteq V$ covers all edges, just choose $S_v$ for all $v \in W$, it covers all $U$.

Set-Cover $f(G, k)$ is yes $\Rightarrow$ Vertex-Cover $(G, k)$ is yes

If $(S_{v_1}, ..., S_{v_k})$ covers all $U$, the set $\{v_1, ..., v_k\}$ covers all edges of $G$. 
A decision problem is a computational problem where the answer is just yes/no.

Here, we study computational complexity of decision Problems.

Why?

- much simpler to deal with
- Decision version is not harder than Search version, so it is easier to lower bound Decision version
- Less important, usually, you can use decider multiple times to find an answer.
Define $P$ (polynomial-time) to be the set of all decision problems solvable by algorithms whose worst-case running time is bounded by some polynomial in the input size.

Do we well understand $P$?
• We can prove that a problem is in $P$ by exhibiting a polynomial time algorithm
• It is in most cases very hard to prove a problem is not in $P$. 
Beyond P?

We have seen many problems that seem hard:

- Independent Set
- 3-coloring
- Min Vertex Cover
- 3-SAT

Given a 3-CNF \((x_1 \lor \overline{x_2} \lor x_9) \land (\overline{x_2} \lor x_3 \lor x_7) \land \ldots\) is there a satisfying assignment?

**Common Property**: If the answer is yes, there is a “short” proof (a.k.a., certificate), that allows you to verify (in polynomial-time) that the answer is yes.

- The proof may be hard to find
Certifier: Algorithm $C(x, t)$ is a certifier for problem $A$ if for every string $x$, the answer is “yes” iff there exists a string $t$ such that $C(x, t) = \text{yes}$.

Intuition: Certifier doesn't determine whether answer is “yes” on its own; rather, it checks a proposed proof $t$ that answer is “yes”.

NP: Decision problems for which there exists a poly-time certifier.

Remark. NP stands for nondeterministic polynomial-time.
Example: 3SAT is in NP

Given a 3-CNF formula, is there a satisfying assignment?

**Certificate:** An assignment of truth values to the n boolean variables.

**Verifier:** Check that each clause has at least one true literal.

**Ex:** $(x_1 \lor \overline{x}_3 \lor x_4) \land (x_2 \lor \overline{x}_4 \lor x_3) \land (x_2 \lor \overline{x}_1 \lor x_3)$

Certificate: $x_1 = T, x_2 = F, x_3 = T, x_4 = F$

**Conclusion:** 3-SAT is in NP
What do we know about NP?

- Nobody knows if all problems in NP can be done in polynomial time, i.e. does $P = NP$?
  - one of the most important open questions in all of science.
  - Huge practical implications specially if answer is yes

- Every problem in $P$ is in NP
  - one doesn’t even need a certificate for problems in $P$ so just ignore any hint you are given

- Every problem in NP is in exponential time

- Some problems in NP seem really hard
  - nobody knows how to prove that they are really hard to solve, i.e. $P \neq NP$
NP Completeness

**Complexity Theorists Approach:** We don’t know how to prove any problem in NP is hard. So, let’s find hardest problems in NP.

**NP-hard:** A problem B is NP-hard iff for any problem $A \in NP$, we have $A \leq_p B$

**NP-Completeness:** A problem B is NP-complete iff B is NP-hard and $B \in NP$.

**Motivations:**
- If $P \neq NP$, then every NP-Complete problems is not in P. So, we shouldn’t try to design Polytime algorithms
- To show $P = NP$, it is enough to design a polynomial time algorithm for just one NP-complete problem.
Cook-Levin Theorem

Theorem (Cook 71, Levin 73): 3-SAT is NP-complete, i.e., for all problems \( A \in NP \), \( A \leq_p 3\text{-SAT} \).

- So, 3-SAT is the hardest problem in NP.

What does this say about other problems of interest? Like Independent set, Vertex Cover, …

Fact: If \( A \leq_p B \) and \( B \leq_p C \) then, \( A \leq_p C \)

Pf idea: Just compose the reductions from A to B and B to C

So, if we prove 3-SAT \( \leq_p \) Independent set, then Independent Set, Clique, Vertex cover, Set cover are all NP-complete

- 3-SAT \( \leq_p \) Independent Set \( \leq_p \) Vertex Cover \( \leq_p \) Set Cover
3-SAT $\leq_p$ Independent Set

Map a 3-CNF to $(G,k)$. Say $m$ is number of clauses
- Create a vertex for each literal
- Joint two literals if
  - They belong to the same clause (blue edges)
  - The literals are negations, e.g., $x_i, \overline{x_i}$ (red edges)
- Set $k=m$

$$(x_1 \lor \overline{x_3} \lor x_4) \land (x_2 \lor \overline{x_4} \lor x_3) \land (x_2 \lor \overline{x_1} \lor x_3)$$

Polynomial-Time Reduction
Correctness of 3-SAT \( \leq_p \) Indep Set

F satisfiable \( \Rightarrow \) An independent of size m
Given a satisfying assignment, Choose one node from each clause where the literal is satisfied

\[(x_1 \lor \overline{x_3} \lor x_4) \land (x_2 \lor \overline{x_4} \lor x_3) \land (x_2 \lor \overline{x_1} \lor x_3)\]

Satisfying assignment: \( x_1 = T, x_2 = F, x_3 = T, x_4 = F \)

- S has exactly one node per clause \( \Rightarrow \) No blue edges between S
- S follows a truth-assignment \( \Rightarrow \) No red edges between S
- S has one node per clause \( \Rightarrow |S| = m \)
Correctness of $3\text{-SAT} \leq_p \text{Indep Set}$

An independent set of size $m \Rightarrow$ A satisfying assignment

Given an independent set $S$ of size $m$.

$S$ has exactly one vertex per clause (because of blue edges)

$S$ does not have $x_i, \overline{x_i}$ (because of red edges)

So, $S$ gives a satisfying assignment

Satisfying assignment: $x_1 = F, x_2 = ?, x_3 = T, x_4 = T$

$$(x_1 \lor \overline{x_3} \lor x_4) \land (x_2 \lor \overline{x_4} \lor x_3) \land (x_2 \lor \overline{x_1} \lor x_3)$$
Summary

• If a problem is NP-hard it does not mean that all instances are hard, e.g., Vertex-cover has a polynomial-time algorithm in trees.

• We learned the crucial idea of polynomial-time reduction. This can be even used in algorithm design, e.g., we know how to solve max-flow so we reduce image segmentation to max-flow.

• NP-Complete problems are the hardest problem in NP.

• NP-hard problems may not necessarily belong to NP.

• Polynomial-time reductions are transitive relations.