Chapter 8
NP and Computational Intractability

8.3 Definition of NP

Decision Problems

- Decision problem.
  - X is a set of strings.
  - Instance: string s.
  - Algorithm A solves problem X:  \( A(s) = \text{yes} \) iff \( s \in X \).

Polynomial time. Algorithm A runs in poly-time if for every string s, A(s) terminates in at most \( p(|s|) \) "steps", where \( p(\cdot) \) is some polynomial.

PRIMES: \( X = \{2, 3, 5, 7, 11, 13, 17, 23, 29, 31, 37, \ldots\} \)
Algorithm. [Agrawal-Kayal-Saxena, 2002] \( p(|s|) = |s|^8 \).

Definition of P

- Decision problems for which there is a poly-time algorithm.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Description</th>
<th>Algorithm</th>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>MULTIPLE</td>
<td>Is x a multiple of y?</td>
<td>Grade school division</td>
<td>51, 17</td>
<td>51, 16</td>
</tr>
<tr>
<td>RELPRIME</td>
<td>Are x and y relatively prime?</td>
<td>Euclid (300 BCE)</td>
<td>34, 39</td>
<td>34, 51</td>
</tr>
<tr>
<td>PRIMES</td>
<td>Is x prime?</td>
<td>AKS (2002)</td>
<td>53</td>
<td>51</td>
</tr>
<tr>
<td>EDIT-DISTANCE</td>
<td>Is the edit distance between x and y less than 5?</td>
<td>Dynamic programming</td>
<td>neither</td>
<td>accept, reject</td>
</tr>
<tr>
<td>LSOLVE</td>
<td>Is there a vector x that satisfies ( Ax = b )?</td>
<td>Gaussian-Edmonds elimination</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Certification algorithm intuition.
- Certifier views things from "managerial" viewpoint.
- Certifier doesn’t determine whether \( s \in X \) on its own; rather, it checks a proposed proof \( t \) that \( s \in X \).

**Def.** Algorithm \( C(s, t) \) is a **certifier** for problem \( X \) if for every string \( s \), \( s \in X \) iff there exists a string \( t \) such that \( C(s, t) = \text{yes} \).

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

**Remark.** NP stands for **nondeterministic** polynomial-time.

**Certifiers and Certificates: 3-Satisfiability**

**SAT.** Given a CNF formula \( \Phi \), is there a satisfying assignment?

**Certificate.** An assignment of truth values to the \( n \) boolean variables.

**Certifier.** Check that each clause in \( \Phi \) has at least one true literal.

**Ex.**
\[
\left( \bar{x}_1 \lor x_3 \lor x_4 \right) \land \left( x_1 \lor \bar{x}_2 \lor x_3 \right) \land \left( x_1 \lor x_2 \lor x_3 \right) \land \left( \bar{x}_4 \lor \bar{x}_7 \lor \bar{x}_1 \right)
\]

**Instance** \( \bar{x}_1 = 1, \bar{x}_2 = 1, x_3 = 0, x_4 = 1 \)

**Conclusion.** SAT is in NP.

**Certifiers and Certificates:  Composite**

**COMPOSITES.** Given an integer \( s \), is \( s \) composite?

**Certificate.** A nontrivial factor \( t \) of \( s \). Note that such a certificate exists iff \( s \) is composite. Moreover \( |t| \leq |s| \).

**Certifier.**
```java
boolean C(s, t) {
    if (t <= 1 || t >= s)
        return false;
    else if (s is a multiple of t)
        return true;
    else
        return false;
}
```

**Instance.** \( s = 437,669 \).

**Certificate.** \( t = 541 \) or 809.

**Conclusion.** COMPOSITES is in NP.

**Certifiers and Certificates: Hamiltonian Cycle**

**HAM-CYCLE.** Given an undirected graph \( G = (V, E) \), does there exist a simple cycle \( C \) that visits every node?

**Certificate.** A permutation of the \( n \) nodes.

**Certifier.** Check that the permutation contains each node in \( V \) exactly once, and that there is an edge between each pair of adjacent nodes in the permutation.

**Conclusion.** HAM-CYCLE is in NP.
P, NP, EXP

P. Decision problems for which there is a poly-time algorithm.

NP. Decision problems for which there is a poly-time certificate.

EXP. Decision problems for which there is an exponential-time algorithm.

Claim. P ⊆ NP.
Pf. Consider any problem X in P.

• By definition, there exists a poly-time algorithm A(s) that solves X.

• Certificate: t = ε, certifier C(s, t) = A(s).

Claim. NP ⊆ EXP.
Pf. Consider any problem X in NP.

• By definition, there exists a poly-time certificate C(s, t) for X.

• To solve input s, run C(s, t) on all strings t with |t| ≤ p(|s|).

• Return yes, if C(s, t) returns yes for any of these.

The Main Question: P Versus NP

Does P = NP? [Cook 1971, Edmonds, Levin, Yablonski, Gödel]

• Is the decision problem as easy as the certification problem?

• Clay $1 million prize.

Consensus opinion on P = NP? Probably no.

8.4 NP-Completeness
Polynomial Transformation

**Def.** Problem X **polynomial reduces** (Cook) to problem Y if arbitrary instances of problem X can be solved using:
- Polynomial number of standard computational steps, plus
- Polynomial number of calls to oracle that solves problem Y.

**Def.** Problem X **polynomial transforms** (Karp) to problem Y if given any input x to X, we can construct an input y such that x is a yes instance of X iff y is a yes instance of Y.

**Note.** Polynomial transformation is polynomial reduction with just one call to oracle for Y, exactly at the end of the algorithm for X. Almost all previous reductions were of this form.

**Open question.** Are these two concepts the same?

NP-Complete

**NP-complete.** A problem Y in NP with the property that for every problem X in NP, X \( \leq_p Y \).

**Theorem.** Suppose Y is an NP-complete problem. Then Y is solvable in poly-time iff P = NP.

**Pf.** ⇒ If P = NP then Y can be solved in poly-time since Y is in NP.

**Pf.** ⇐ Suppose Y can be solved in poly-time.
- Let X be any problem in NP. Since X \( \leq_p Y \), we can solve X in poly-time. This implies NP \( \subseteq \) P.
- We already know P \( \subseteq \) NP. Thus P = NP.

**Fundamental question.** Do there exist "natural" NP-complete problems?

Circuit Satisfiability

**CIRCUIT-SAT.** Given a combinational circuit built out of AND, OR, and NOT gates, is there a way to set the circuit inputs so that the output is 1?

![Circuit Diagram]

**The "First" NP-Complete Problem

**Theorem.** CIRCUIT-SAT is NP-complete. [Cook 1971, Levin 1973]

**Pf.** (sketch)
- Any algorithm that takes a fixed number of bits \( n \) as input and produces a yes/no answer can be represented by such a circuit. Moreover, if algorithm takes poly-time, then circuit is of poly-size.

- Consider some problem X in NP. It has a poly-time certifier \( C(s, t) \). To determine whether \( s \) is in X, need to know if there exists a certificate \( t \) of length \( p(|s|) \) such that \( C(s, t) = \text{yes} \).
- View \( C(s, t) \) as an algorithm on \( |s| + p(|s|) \) bits (input \( s \), certificate \( t \)) and convert it into a poly-size circuit \( K \).
  - first \(|s|\) bits are hard-coded with \( s \)
  - remaining \( p(|s|) \) bits represent bits of \( t \)
- Circuit \( K \) is satisfiable iff \( C(s, t) = \text{yes} \).
Example

Ex. Construction below creates a circuit K whose inputs can be set so that K outputs true iff graph G has an independent set of size 2.

![Diagram of graph G with vertices and edges]

3-SAT is NP-Complete

Theorem. 3-SAT is NP-complete.

Pf. Suffices to show that CIRCUIT-SAT \( \leq_p \) 3-SAT since 3-SAT is in NP.

- Let K be any circuit.
- Create a 3-SAT variable \( x_i \) for each circuit element i.
- Make circuit compute correct values at each node:
  - \( x_2 = \neg x_3 \) \( \Rightarrow \) add 2 clauses: \( \neg x_3 \lor x_2, \neg x_2 \lor x_3 \)
  - \( x_4 = x_2 \lor x_5 \) \( \Rightarrow \) add 3 clauses: \( x_2 \lor x_5, x_4 \lor x_2, x_4 \lor x_5 \)
  - \( x_6 = x_2 \lor x_3 \) \( \Rightarrow \) add 3 clauses: \( x_2 \lor x_3, x_2 \lor x_6, x_3 \lor x_6 \)

- Hard-coded input values and output value.
  - \( x_0 = 0 \) \( \Rightarrow \) add 1 clause: \( x_0 \)
  - \( x_0 = 1 \) \( \Rightarrow \) add 1 clause: \( \neg x_0 \)

- Final step: turn clauses of length < 3 into clauses of length exactly 3.

Establishing NP-Completeness

Remark. Once we establish first "natural" NP-complete problem, others fall like dominoes.

Recipe to establish NP-completeness of problem Y.

1. Show that Y is in NP.
2. Choose an NP-complete problem X.
3. Prove that X \( \leq_p \) Y.

Justification. If X is an NP-complete problem, and Y is a problem in NP with the property that X \( \leq_p \) Y then Y is NP-complete.

Pf. Let W be any problem in NP. Then W \( \leq_p X \) \( \leq_p Y \).

- By transitivity, W \( \leq_p Y \).
- Hence Y is NP-complete.

NP-Completeness

Observation. All problems below are NP-complete and polynomial reduce to one another!
Some NP-Complete Problems

Six basic genres of NP-complete problems and paradigmatic examples.

- Packing problems: SET-PACKING, INDEPENDENT SET.
- Covering problems: SET-COVER, VERTEX-COVER.
- Constraint satisfaction problems: SAT, 3-SAT.
- Sequencing problems: HAMILTONIAN-CYCLE, TSP.
- Partitioning problems: 3D-MATCHING, 3-COLOR.
- Numerical problems: SUBSET-SUM, KNAPSACK.

Practice. Most NP problems are either known to be in P or NP-complete.

Notable exceptions. Factoring, graph isomorphism, Nash equilibrium.

Extent and Impact of NP-Completeness

Extent of NP-completeness. [Papadimitriou 1995]
- Prime intellectual export of CS to other disciplines.
- More than "compiler", "operating system", "database"
- Broad applicability and classification power.
- "Captures vast domains of computational, scientific, mathematical endeavors, and seems to roughly delimit what mathematicians and scientists had been aspiring to compute feasibly."

NP-completeness can guide scientific inquiry.
- 1926: Ising introduces simple model for phase transitions.
- 1944: Onsager solves 2D case in tour de force.
- 19xx: Feynman and other top minds seek 3D solution.

More Hard Computational Problems

Aerospace engineering: optimal mesh partitioning for finite elements.
Biology: protein folding.
Chemical engineering: heat exchanger network synthesis.
Civil engineering: equilibrium of urban traffic flow.
Economics: computation of arbitrage in financial markets with friction.
Electrical engineering: VLSI layout.
Environmental engineering: optimal placement of contaminant sensors.
Financial engineering: find minimum risk portfolio of given return.
Game theory: find Nash equilibrium that maximizes social welfare.
Genomics: phylogeny reconstruction.
Mechanical engineering: structure of turbulence in sheared flows.
Medicine: reconstructing 3-D shape from biplane angiogram.
Operations research: optimal resource allocation.
Physics: partition function of 3-D Ising model in statistical mechanics.
Politics: Shapley-Shubik voting power.
Pop culture: Minesweeper consistency.
Statistics: optimal experimental design.