8. NP and Computational Intractability

8.1. Polynomial-Time Reductions

Classify Problems According to Computational Requirements

Q. Which problems will we be able to solve in practice?

A working definition. [Cobham 1964, Edmonds 1965, Rabin 1966]
Those with polynomial-time algorithms.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Yes</th>
<th>Probably no</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shortest path</td>
<td></td>
<td>Longest path</td>
</tr>
<tr>
<td>Euler cycle</td>
<td></td>
<td>Hamiltonian cycle</td>
</tr>
<tr>
<td>Min cut</td>
<td></td>
<td>Max cut</td>
</tr>
<tr>
<td>2-SAT</td>
<td></td>
<td>3-SAT</td>
</tr>
<tr>
<td>Planar 4-color</td>
<td></td>
<td>Planar 3-color</td>
</tr>
<tr>
<td>Bipartite vertex cover</td>
<td></td>
<td>Vertex cover</td>
</tr>
<tr>
<td>Matching</td>
<td></td>
<td>3D-matching</td>
</tr>
<tr>
<td>Primality testing</td>
<td></td>
<td>Factoring</td>
</tr>
</tbody>
</table>
Classify Problems

**Desiderata.** Classify problems according to those that can be solved in polynomial-time and those that cannot.

**Provably requires exponential-time.**
- Given a Turing machine, does it halt in at most $k$ steps?
- Given a board position in an $n$-by-$n$ generalization of chess, can black guarantee a win?

**Bad news.** Huge number of fundamental problems have defied classification for decades.

**Worse news.** Many were shown to be "computationally equivalent" and intractable for all practical purposes.

Polynomial-Time Reduction

**Purpose.** Classify problems according to relative difficulty.

**Design algorithms.** If $X \leq_p Y$ and $Y$ can be solved in polynomial-time, then $X$ can also be solved in polynomial time.

**Establish intractability.** If $X \leq_Y Y$ and $X$ cannot be solved in polynomial-time, then $Y$ cannot be solved in polynomial time.

**Establish equivalence.** If $X \leq_p Y$ and $Y \leq_p X$, we use notation $X =_p Y$.

---

Polynomial-Time Reduction

**Desiderata'.** Suppose we could solve $X$ in polynomial-time. What else could we solve in polynomial time?

**Reduction.** Problem $X$ polynomial reduces 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 $Y$. [1]

**Notation.** $X \leq_p Y$.

**Remarks.**
- We pay for time to write down instances sent to black box $\Rightarrow$ instances of $Y$ must be of polynomial size.
- Note: Cook reducibility. [2] in contrast to Karp reductions

---

### Basic strategies.
- Reduction by simple equivalence.
- Reduction from special case to general case.
- Reduction by encoding with gadgets.
**Independent Set**

**INDEPENDENT SET**: Given a graph $G = (V, E)$ and an integer $k$, is there a subset of vertices $S \subseteq V$ such that $|S| \geq k$, and for each edge at most one of its endpoints is in $S$?

**Ex.** Is there an independent set of size $\geq 6$? Yes.
**Ex.** Is there an independent set of size $\geq 7$? No.

**Vertex Cover**

**VERTEX COVER**: Given a graph $G = (V, E)$ and an integer $k$, is there a subset of vertices $S \subseteq V$ such that $|S| = k$, and for each edge, at least one of its endpoints is in $S$?

**Ex.** Is there a vertex cover of size $\leq 4$? Yes.
**Ex.** Is there a vertex cover of size $\leq 3$? No.

---

**Claim.** VERTEX-COVER $\#P$ INDEPENDENT-SET.

**Pf.** We show $S$ is an independent set iff $V \setminus S$ is a vertex cover.

- Let $S$ be any independent set.
  - Consider an arbitrary edge $(u, v)$.
  - $u \notin S$ or $v \notin S$.
  - Thus, $V \setminus S$ covers $(u, v)$.

- Let $V \setminus S$ be any vertex cover.
  - Consider two nodes $u \in S$ and $v \in S$.
  - $(u, v) \notin E$ since $V \setminus S$ is a vertex cover.
  - Thus, no two nodes in $S$ are joined by an edge $\Rightarrow S$ independent set.
Polynomial-Time Reduction

Basic strategies.
- Reduction by simple equivalence.
- Reduction from special case to general case.
- Reduction by encoding with gadgets.

Set Cover

**SET COVER**: Given a set \( U \) of elements, a collection \( S_1, S_2, \ldots, S_m \) of subsets of \( U \), and an integer \( k \), does there exist a collection of \( \leq k \) of these sets whose union is equal to \( U \)?

Sample application.
- \( m \) available pieces of software.
- \( U \) of \( n \) capabilities that we would like our system to have.
- The \( i \)th piece of software provides the set \( S_i \subseteq U \) of capabilities.
- Goal: achieve all \( n \) capabilities using fewest pieces of software.

**Ex:**
\[
U = \{1, 2, 3, 4, 5, 6, 7\} \\
k = 2 \\
S_1 = \{3, 7\} \\
S_2 = \{2, 4\} \\
S_3 = \{3, 4, 5, 6\} \\
S_4 = \{5\} \\
S_5 = \{1\} \\
S_6 = \{1, 2, 6, 7\}
\]

Vertex Cover Reduces to Set Cover

**Claim.** \( \text{VERTEX-COVER} \leq_p \text{SET-COVER} \).

**Pf.** Given a \( \text{VERTEX-COVER} \) instance \( G = (V, E) \), \( k \), we construct a set cover instance whose size equals the size of the vertex cover instance.

**Construction.**
- \( k = k \), \( U = E \), \( S_v = \{e \in E : e \text{ incident to } v\} \)
- Set-cover of size \( \leq k \) iff vertex cover of size \( \leq k \).

**Integer Programming**

**INTEGER-PROGRAMMING**: Given integers \( a_{ij} \) and \( b_i \), find integers \( x_j \) that satisfy:
\[
\sum_{j=1}^{m} a_{ij} x_j \geq b_i \quad 1 \leq i \leq m \\
x_j \geq 0 \quad 1 \leq j \leq n \\
x_j \text{ integral } 1 \leq j \leq n
\]

**Claim.** \( \text{VERTEX-COVER} \leq_p \text{INTEGER-PROGRAMMING} \).
Polynomial-Time Reduction

Basic strategies.
- Reduction by simple equivalence.
- Reduction from special case to general case.
- Reduction by encoding with gadgets.

Satisfiability

Literal: A Boolean variable or its negation. $x_i$ or $\overline{x}_i$

Clause: A disjunction of literals. $C_j = x_1 \lor x_2 \lor x_3$

Conjunctive normal form: A propositional formula $\Phi$ that is the conjunction of clauses.

$\Phi = C_1 \land C_2 \land C_3 \land C_4$

SAT: Given CNF formula $\Phi$, does it have a satisfying truth assignment?

3-SAT: SAT where each clause contains exactly 3 literals.

**Claim.** $3\text{-SAT} \leq_p \text{INDDEPENDENT-SET}$.

**Pf.** Given an instance $\Phi$ of 3-SAT, we construct an instance $(G, k)$ of INDDEPENDENT-SET that has an independent set of size $k$ if and only if $\Phi$ is satisfiable.

**Construction.**
- $G$ contains 3 vertices for each clause, one for each literal.
- Connect 3 literals in a clause in a triangle.
- Connect literal to each of its negations.

Given a CNF formula $\Phi = (x_1 \lor x_2 \lor x_3) \land (x_1 \lor x_2 \lor x_4) \land (\overline{x}_1 \lor x_2 \lor x_4)$, we construct the following graph $G$ for $k = 3$:

- Vertices: $x_1, x_2, x_3, x_4$.
- Edges: $x_1 \leftrightarrow x_2 \leftrightarrow x_3 \leftrightarrow x_4$.

3 Satisfiability Reduces to Independent Set

**Claim.** $G$ contains independent set of size $k = |\Phi|$ if and only if $\Phi$ is satisfiable.

**Pf.** $\Rightarrow$ Let $S$ be an independent set of size $k$.
- $S$ must contain exactly one vertex in each triangle.
- Set these literals to true. All other variables are set consistently.
- Truth assignment is consistent and all clauses are satisfied.

$\Leftarrow$ Given satisfying assignment, select one true literal from each triangle. This is an independent set of size $k$. $\blacksquare$
Review

Basic reduction strategies.
- Simple equivalence: INDEPENDENT-SET $\leq_P$ VERTEX-COVER.
- Special case to general case: VERTEX-COVER $\leq_P$ SET-COVER.
- Encoding with gadgets: 3-SAT $\leq_P$ INDEPENDENT-SET.

Transitivity. If $X \leq_P Y$ and $Y \leq_P Z$, then $X \leq_P Z$.

Pf idea. Compose the two algorithms.

Ex: 3-SAT $\leq_P$ INDEPENDENT-SET $\leq_P$ VERTEX-COVER $\leq_P$ SET-COVER.

Self-Reducibility

Decision problem. Does there exist a vertex cover of size $\leq k$?
Search problem. Find vertex cover of minimum cardinality.

Self-reducibility. Search problem $\leq_P$ decision version.
- Applies to all (NP-complete) problems in this chapter.
- Justifies our focus on decision problems.

Ex: to find min cardinality vertex cover.
- (Binary) search for cardinality $k^*$ of min vertex cover.
- Find a vertex $v$ such that $G - \{v\}$ has a vertex cover of size $\leq k^* - 1$.
  - any vertex in any min vertex cover will have this property
- Include $v$ in the vertex cover.
- Recursively find a min vertex cover in $G - \{v\}$.
  \[
  \text{delete } v \text{ and all incident edges}
  \]