Chapter 10
Part 1: Reduction

Polynomial-Time Reduction

If \( X \leq_P Y \) and the code of \( Y \) is \( B \), we may obtain the code \( A \) which uses \( B \) to solve \( X \) (the time spent by \( B \) is not cared).

Design algorithms. If \( X \leq_P Y \) and \( Y \) can be solved in polynomial-time, then \( X \) can also be solved in polynomial time. That is, if \( Y \) is easy, so is \( X \).

Establish equivalence. If \( X \leq_P Y \) and \( Y \leq_P X \), we use notation \( X \equiv_P Y \).

E.g., Max-Bipartite-Matching \( \equiv_P \) Min-Bipartite-Vertex-Cover

How many times did you find a solution of size \( \geq 7 \)?

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.

Clique

CLIQUE: 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 pair \((x, y)\) of points in \( S \), \((x, y)\) is an edge of \( E \)?

Claim. CLIQUE \( \equiv_P \) INDEPENDENT-SET.
Pf. We show \( S \) is an independent set iff \( S \) is a clique of \( G' \), where \( G' \) is the complement of \( G \) if \( G \) is a clique of \( G' \).

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 \( |S| \leq k \), and for each edge, at least one of its endpoints is in \( S \)?

Ex. Is there a vertex cover of size \( \leq 47 \)? Yes.
Ex. Is there a vertex cover of size \( \leq 37 \)? No.
**Vertex Cover and Independent Set**

Claim. VERTEX-COVER $\equiv$ INDEPENDENT-SET.

Pf. We show $S$ is an independent set iff $V - S$ is a vertex cover. Consequently, $S$ is a maximum independent set iff $V - S$ is a minimum vertex cover. If we have an efficient algorithm to solve one, we will have efficient algorithm to solve the other.

**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.
- Set $U$ 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:

$U = \{1, 2, 3, 4, 5, 6, 7\}$
$k = 2$
$S_1 = \{3, 7\}$
$S_2 = \{3, 4, 5, 6\}$
$S_3 = \{1\}$
$S_4 = \{1, 2, 6, 7\}$
$S_5 = \{5\}$

**3-Satisfiability**

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

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

Clause: A disjunction of literals. $C_i = x_i \lor x_j \lor x_k$

Conjunctive normal form: A propositional formula $\Phi$ that is the conjunction of clauses. $\Phi = C_1 \land C_2 \land C_3 \land \ldots C_n$

3-SAT: A SAT formula in conjunctive normal form where each clause contains exactly 3 literals.

Ex:

$(x_1 \lor x_2 \lor x_3) \land (x_4 \lor x_5 \lor x_6) \land (x_1 \lor x_2 \lor \overline{x_3})$

Yes: $x_1$ true, $x_2$ true, $x_3$ false.

**Vertex Cover and Independent Set**

Claim. VERTEX-COVER $\equiv$ INDEPENDENT-SET.

Pf. We show $S$ is an independent set iff $V - S$ is a vertex cover.

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

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

**Vertex Cover and Independent Set**

Claim. VERTEX-COVER $\leq$ SET-COVER.

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

Construction.
- Create SET-COVER instance:
  - $k = k$, $U = E$, $S_v = \{e \in E : e \text{ incident to } v\}$
- Set-cover of size $\leq k$ iff vertex cover of size $\leq k$.

**Set Cover**

$S = \{3, 7\}$ $S_1 = \{2, 4\}$
$S_2 = \{3, 4, 5, 6\}$ $S_3 = \{5\}$
$S_4 = \{1\}$ $S_5 = \{1, 2, 6, 7\}$

**3-SAT is as hard as SAT**

Claim. SAT $\leq$ 3-SAT.

Pf. Let $K$ be any circuit representing the formula of SAT.
- Create a 3-SAT variable $x_i$ for each circuit element $i$.
- Make circuit compute correct values at each node:
  - $x_2 = \overline{x_3}$ $\Rightarrow$ add 2 clauses: $x_1 \lor \overline{x_2} \lor \overline{x_3}$
  - $x_1 = x_4 \lor x_5$ $\Rightarrow$ add 3 clauses: $x_1 \lor x_4 \lor \overline{x_5}$
  - $x_0 = x_1 \land x_2$ $\Rightarrow$ add 3 clauses: $x_0 \land x_1 \land x_2$ $\lor \overline{x_0} \land \overline{x_1} \land \overline{x_2}$
- Hard-coded input values and output value.
  - $x_4 = 0$ $\Rightarrow$ add 1 clause: $\overline{x_4}$
  - $x_0 = 1$ $\Rightarrow$ add 1 clause: $x_0$
- Final step: turn clauses of length < 3 into clauses of length exactly 3 by introducing new variables.
Summary

Basic reduction strategies.
- Simple equivalence: INDEPENDENT-SET \equiv P VERTEX-COVER
- Special case to general case: VERTEX-COVER \leq P SET-COVER, or SAT \leq P 3-SAT.

Transitivity. If X \leq P Y and Y \leq P Z, then X \leq P Z.

Pf idea. Compose the two algorithms.

Ex: INDEPENDENT-SET \leq P VERTEX-COVER \leq P SET-COVER.

So INDEPENDENT-SET \leq P SET-COVER

For the clique problem:

- Decision Problem:
  - Input: a graph G = (V, E), integer k.
  - Question: Does G have a clique of size \geq k?
- Optimization problem: Find a clique of maximum cardinality.

In general, Decision problem is easier than Optimization problem. However, the difference is by a polynomial reduction.

Decision vs Optimization

For the clique problem, suppose we have an algorithm A to answer the decision problem: A(G, k) = yes iff G has a clique of size k. How can we find the maximum clique of G?

- Step 1: Decide k, the size of the maximum clique:
  for i from n downto 1, if A(G, i) = yes and A(G, i+1) = no, then return i;
- Step 2: Decide the actual max clique:
  for each vertex v in G
    if (A(G - v, k) = yes) G = G - v;
    // G – v means v is deleted from G.

- Suppose we have an algorithm B which returns the maximum clique of G. How can we solve the decision version of the clique problem?

Vertex Cover Problem

- Decision Problem:
  - Input: a graph G = (V, E), integer k.
  - Question: Does G have a vertex cover of size \leq k?
- Optimization problem. Find vertex cover of minimum cardinality of G.

Self-reducibility: Decision and Optimization Problems are all equivalent (=P)
Applies to all (NP-complete) problems.
Justifies our focus on decision problems.

What is Linear Programming?

Toy manufacturer can produce skateboards and dolls. Both require the precious resource of plastic, of which there are 60 units available. Skateboards take five units of plastic and make $1 profit. Dolls take two units of plastic and make $0.55 profit.

What is the number of dolls and skateboards the company can produce to maximize profit?
What is Linear Programming?
A mathematical tool for maximizing or minimizing a quantity (usually profit or cost of production), subject to certain constraints.

Of all computations and decisions made by management in business, 50-90% of those involve linear programming.

Setting Up Problems
First identify components of the problem:
1. Resources – Plastic (60 available)
2. Products – Skateboards & Dolls
3. Recipes – Skateboards (5 units), Dolls (2 units)
4. Profits – Skateboards ($1.00), Dolls ($0.55)
5. Objective – Maximize profit

Second, make a mixture chart:

Translate Mixture Chart into Formulas

2 Groups of Equations:
- Objective Equation (profit equation)
- Constraints (minimum constraints, resource constraints...)

Objective Equation – total profit given number of units produced
\[ P = 1x + 0.55y \]
Constraints – usually inequalities
\[ 5x + 2y \leq 60 \]

Linear Programming
Feasible Region – region which consists of all possible solution choices for a particular problem

Using the constraint equation we get the following graph:

Corner Point Principle
Which point is optimal?
• Any point in feasible region will satisfy constraint equation, but which will maximize profit equation?

Corner Point Principle
In LP problem, if a maximal value exists, one of corner points on feasible region should have it.
Types of Integer Programming Models

A linear program in which all the variables are restricted to be integers is called an integer linear program (ILP). If only a subset of the variables are restricted to be integers, the problem is called a mixed integer linear program (MILP). Binary variables are variables whose values are restricted to be 0 or 1. If all variables are restricted to be 0 or 1, the problem is called a 0-1 or binary integer program.

Example: All-Integer LP

Consider the following all-integer linear program:

\[
\begin{align*}
\text{Max} & \quad 3x_1 + 2x_2 \\
\text{s.t.} & \quad 3x_1 + x_2 \leq 9 \\
& \quad x_1 + 3x_2 \leq 7 \\
& \quad -x_1 + x_2 \leq 1 \\
& \quad x_1, x_2 \geq 0 \text{ and integer}
\end{align*}
\]

Example: All-Integer LP

LP Relaxation

Solving the problem as a linear program ignoring the integer constraints, the optimal solution to the linear program gives fractional values for both \(x_1\) and \(x_2\). From the graph on the previous slide, we see that the optimal solution to the linear program is:

\(x_1 = 2.5, \ x_2 = 1.5, \ z = 10.5\)

Example: All-Integer LP

Rounding Up

If we round up the fractional solution \((x_1 = 2.5, \ x_2 = 1.5)\) to the LP relaxation problem, we get \(x_1 = 3\) and \(x_2 = 2\). From the graph on the next page, we see that this point lies outside the feasible region, making this solution infeasible.
Example: All-Integer LP

Rounding Down

By rounding the optimal solution down to \( x_1 = 2, x_2 = 1 \), we see that this solution indeed is an integer solution within the feasible region, and substituting in the objective function, it gives \( z = 8 \).

We have found a feasible all-integer solution, but have we found the optimal all-integer solution?

The answer is NO! The optimal solution is \( x_1 = 3 \) and \( x_2 = 0 \) giving \( z = 9 \), as evidenced in the next two slides.

Example: All-Integer LP

Complete Enumeration of Feasible ILP Solutions

There are eight feasible integer solutions to this problem:

\[
\begin{array}{cccc}
1 & 0 & 0 & 0 \\
1 & 1 & 0 & 3 \\
2 & 0 & 6 & 6 \\
3 & 0 & 9 & 8 \\
4 & 1 & 1 & 5 \\
6 & 1 & 1 & 5 \\
7 & 2 & 1 & 8 \\
8 & 1 & 2 & 7 \\
\end{array}
\]

Decision Problem: 0-1 Programming

**0-1 PROGRAMMING**. Given a \( n \) by \( m \) matrix \( A \), a vector \( B \) of \( m \) numbers, a vector \( X \) of \( n \) variables, is there a binary solution of \( X \) such that \( AX \leq B \)?

Claim: 3-SAT \( \leq_p \) 0-1 PROGRAMMING.

Pf.