**NP and Computational Intractability**

**Chapter 8**

**Directed Hamiltonian Cycle**

**Claim.** \( G \) has a Hamiltonian cycle iff \( G' \) does.

**Pf.**

1. Suppose \( G \) has a directed Hamiltonian cycle \( \Gamma \).
   - Then \( G' \) has an undirected Hamiltonian cycle (same order).

2. Suppose \( G' \) has an undirected Hamiltonian cycle \( \Gamma' \).
   - \( \Gamma' \) must visit nodes in \( G' \) using one of the following two orders:
     - \( \ldots, B, R, G, R, B, G, R, B, \ldots \)
     - \( \ldots, B, G, B, R, G, B, R, G, B, \ldots \)
   - Blue nodes in \( \Gamma' \) make up directed Hamiltonian cycle \( \Gamma \) in \( G \), or reverse of one. •

**Traveling Salesman Problem**

Traveling Salesman Problem (TSP): Given a complete graph with nonnegative edge costs, find a minimum cost cycle visiting every vertex exactly once.

Example: Given a number of cities and the costs of traveling from any city to any other city, what is the cheapest round-trip route that visits each city exactly once and then returns to the starting city?

**Claim.** \( HAM-CYCLE \leq_p TSP \).

**Pf.**

Given a graph \( G = (V, E) \), construct a complete weighted graph \( G' = (V, V \times V, W) \), such that \( W(e) = 1 \) for \( e \in E \) and \( W(e) = 2 \) for \( e \not\in E \), and \( d = |V| \).
8.7 Graph Coloring

Basic genres:
- 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.

3-Colorability

**Claim.** 3-SAT ≤ P 3-COLOR.

**Pf.** Given 3-SAT instance Φ, we construct an instance of 3-COLOR that is 3-colorable iff Φ is satisfiable.

**Construction.**
1. For each literal, create a node.
2. Create 3 new nodes T, F, B; connect them in a triangle, and connect each literal to B.
3. Connect each literal to its negation.
4. For each clause, add gadget of 6 nodes and 13 edges.

```
T  B  F
x1 x1 x2 x2 x3 x3
true false
```

3-Colorability

**Claim.** Graph is 3-colorable iff Φ is satisfiable.

**Pf.** Suppose graph is 3-colorable.
1. Let’s call the colors of T and F “true color” and “false color”, resp.
2. Then each literal has either true color or false color.
3. A literal and its negation always have opposite colors.
3-Colorability

Claim. Graph is 3-colorable iff $\Phi$ is satisfiable.

Pf. $\Rightarrow$ Suppose graph is 3-colorable.
   (i) Let's call the colors of T and F "true color" and "false color", resp.
   (ii) Then each literal has either true color or false color.
   (iii) A literal and its negation always have opposite colors.
   (iv) Ensures at least one literal in each clause is true color.

3-Colorability

Claim. Graph is 3-colorable iff $\Phi$ is satisfiable.

Pf. $\Rightarrow$ Suppose graph is 3-colorable.
   (i) Let's call the colors of T and F "true color" and "false color", resp.
   (ii) Then each literal has either true color or false color.
   (iii) A literal and its negation always have opposite colors.
   (iv) Ensures at least one literal in each clause is true color.
**Map 3-Colorability**

**MAP-3-COLOR**: Given a planar map, can it be colored using 3 colors so that no adjacent regions have the same color?

**YES instance.**

**NO instance.**

---

**Planarity**

**Def.** A graph is **planar** if it can be embedded in the plane in such a way that no two edges cross.

**Applications:** VLSI circuit design, computer graphics.

**Kuratowski’s Theorem.** An undirected graph $G$ is non-planar iff it contains a subgraph homeomorphic to $K_5$ or $K_{3,3}$.

---

**Planarity Testing**

**Planarity testing.** [Hopcroft-Tarjan 1974] $O(n)$

**Remark.** Many intractable graph problems can be solved in poly-time if the graph is planar; many tractable graph problems can be solved faster if the graph is planar.
**Planar Graph 3-Colorability**

Q. Is this planar graph 3-colorable?

**Map 3-Colorability and Graph 3-Colorability**

Claim. Map-3-COLOR ≤ \text{P}_{\text{PLANAR-GRAPH-3-COLOR}}.

Pf sketch. Create a vertex for each region, and an edge between regions that share a nontrivial border.

**Planar Graph 3-Colorability**

Claim. W is a planar graph such that:
- In any 3-coloring of W, opposite corners have the same color.
- Any assignment of colors to the corners in which opposite corners have the same color extends to a 3-coloring of W.

Pf. W has only two 3-colorings (or by permuting colors).

four corners same color

four corners two colors

**Planar Graph 3-Colorability**

Claim. 3-COLOR ≤ \text{P}_{\text{PLANAR-GRAPH-3-COLOR}}.

Pf. Given instance of 3-COLOR, draw graph in plane, letting edges cross.
- Replace each edge crossing with planar gadget W.
- In any 3-coloring of W, a ≠ a' and b ≠ b'.
- If a = a' and b = b' then can extend to a 3-coloring of W.
Planar Graph 3-Colorability

Claim. $3\text{-COLOR} \leq_{\text{P}} \text{PLANAR-GRAPH-3-COLOR}$.

Proof. Given instance of $3\text{-COLOR}$, draw graph in plane, letting edges cross.
- Replace each edge crossing with planar gadget $W$.
- In any 3-coloring of $W$, $a \neq a'$ and $b \neq b'$.
- If $a \neq a'$ and $b \neq b'$ then can extend to a 3-coloring of $W$.

8.8 Numerical Problems

Basic genres:
- 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: 3-COLOR, 3D-MATCHING.
- Numerical problems: SUBSET-SUM, Knapsack.

Planar k-Colorability

$\text{PLANAR-2-COLOR}$. Solvable in linear time.

$\text{PLANAR-3-COLOR}$. NP-complete.

$\text{PLANAR-4-COLOR}$. Solvable in O(1) time.

Theorem. [Appel-Haken, 1976] Every planar map is 4-colorable.
- Resolved century-old open problem.
- Used 50 days of computer time to deal with many special cases.
- First major theorem to be proved using computer.

False intuition. If $\text{PLANAR-3-COLOR}$ is hard, then so is $\text{PLANAR-4-COLOR}$ and $\text{PLANAR-5-COLOR}$.

Subset Sum

SUBSET-SUM. Given a set of natural numbers $w_1, w_2, \ldots, w_n$ and an integer $W$, is there a subset that adds up to exactly $W$?

Ex: $\{1, 4, 16, 64, 256, 1040, 1041, 1093, 1284, 1344\}$, $W = 3754$.
Yes: $1 + 16 + 64 + 256 + 1040 + 1093 + 1284 = 3754$.

Remark. With arithmetic problems, input integers are encoded in binary. Polynomial reduction must be polynomial in the size of binary encoding.

Claim. $3\text{-SAT} \leq_{\text{P}} \text{SUBSET-SUM}$.

Proof. Given an instance $\Phi$ of 3-SAT, we construct an instance of SUBSET-SUM that has solution iff $\Phi$ is satisfiable.
Subset Sum

**Construction.** Given 3-SAT instance $\Phi$ with $n$ variables and $k$ clauses, form $2n + 2k$ decimal integers, each of $n+k$ digits, as illustrated below.

Claim. $\Phi$ is satisfiable iff there exists a subset that sums to $W$.

Pf. No carries possible.

$$\begin{align*}
C_1 &= \overline{x} \lor \overline{y} \lor \overline{z} \\
C_2 &= \overline{x} \lor \overline{y} \lor \overline{z} \\
C_3 &= \overline{x} \lor \overline{y} \lor \overline{z} \\
\end{align*}$$

dummies to get clause columns to sum to 4

$$\begin{array}{cccccc}
\text{x} & \text{y} & \text{z} & \text{C}_1 & \text{C}_2 & \text{C}_3 \\
1 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 1 \\
0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 2 & 0 & 0 \\
0 & 0 & 0 & 0 & 2 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 1 \\
2 & 0 & 0 & 0 & 0 & 1 \\
3 & 1 & 1 & 1 & 1 & 1 \\
\end{array}$$

decoded to get clause column to sum to 4

**Bin Packing**

**BIN-PACKING.** Given a set $S$ of real numbers $w_1, \ldots, w_n$, $0 \leq w_i \leq 1$, and integer $K$, is there a partition of $S$ into $K$ subsets such that each subset adds up no more than 1?

Claim. **PARTITION** $\leq P$ **SUBSET-SUM.**

Pf.

**Set Partition**

**PARTITION.** Given a set of natural numbers $w_1, \ldots, w_n$, is there a subset that adds up to exactly half sum of all $w_i$?

Claim. **PARTITION** $\leq P$ **SUBSET-SUM.**

Pf. **PARTITION** is a special of **SUBSET-SUM.**

Claim. **SUBSET-SUM** $\leq P$ **PARTITION.**

Pf.

**The Knapsack Problem**

**Input**
- Capacity $K$
- $n$ items with weights $w_i$ and values $v_i$

**Goal**
- Output a set of items $S$ such that
  - the sum of weights of items in $S$ is at most $K$
  - and the sum of values of items in $S$ is maximized
The Simplest Versions...

Can items be divided up such that only a portion is taken?

The thief can hold 5 pounds and has to choose from:
- 3 pounds of gold dust at $379.22/pound
- 6 pounds of silver dust at $188.89/pound
- 1/9 pound of platinum dust at $433.25/pound

Are all of the weights or total values identical?

The thief breaks into a ring shop where all of the rings weigh 1oz. He can hold 12 ounces; which should he take?

A Deceptively Hard Version...

What if each problem has the same price/pound?

This problem reduces to the bin-packing problem: we want to fit as many pounds of material into the knapsack as possible.

How can we approach this problem?

Example

The thief breaks into a gold refinery; he can steal from a selection of raw gold nuggets, each of the same value per pound. If he can carry 50 pounds, what selection would maximize the amount he carries out?

<table>
<thead>
<tr>
<th>Weight (pounds)</th>
<th>Value (dollars)</th>
</tr>
</thead>
<tbody>
<tr>
<td>47.3</td>
<td>6</td>
</tr>
<tr>
<td>36.7</td>
<td>5.6</td>
</tr>
<tr>
<td>25.5</td>
<td>5.6</td>
</tr>
<tr>
<td>16.7</td>
<td>5.4</td>
</tr>
<tr>
<td>8.8</td>
<td>5.3</td>
</tr>
</tbody>
</table>

An Easier Version...

What if all of the sizes we are working with are relatively small integers? For example, if we could fit 10 pounds and:

- Object A is 2 pounds and worth $40
- Object B is 3 pounds and worth $50
- Object C is 1 pound and worth $100
- Object D is 5 pounds and worth $95
- Object E is 3 pounds and worth $30

We can use dynamic programming!
Defining subproblems

Define \( P(i, w) \) to be the problem of choosing a set of objects from the first \( i \) objects that maximizes value subject to weight constraint of \( w \).

\( V(i, w) \) is the value of this set of items

Original problem corresponds to \( V(n, K) \)

Recursive Definition:

\[
V(i, w) = \max (V(i-1, w-w_i) + v_i, V(i-1, w))
\]

- A maximal solution for \( P(i, w) \) either
  - uses item \( i \) (first term in max)
  - or does NOT use item \( i \) (second term in max)

\( V(0, w) = 0 \) (no items to choose from)

\( V(i, 0) = 0 \) (no weight allowed)

Decision Problem: Knapsack

**KNAPSACK**. Given a set of \( n \) objects, each object \( i \) has a weight \( w_i \) and value \( v_i \), and two numbers \( W \) and \( V \), is there a subset of objects whose total weight is no more than \( W \) and whose total value is no less than \( V \)?

**Claim.** PARTITION \( \leq \) \( \_ \_ \) KNAPSACK.

**Pf.**

The solution...

<table>
<thead>
<tr>
<th>Items</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>( w_A = 2 )</td>
<td>( v_A = $40 )</td>
<td>( w_B = 3 )</td>
<td>( v_B = $50 )</td>
<td>( w_C = 1 )</td>
<td>( v_C = $100 )</td>
</tr>
<tr>
<td>( w_D = 5 )</td>
<td>( v_D = $95 )</td>
<td>( w_E = 3 )</td>
<td>( v_E = $30 )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1</th>
<th>0</th>
<th>0</th>
<th>$100</th>
<th>$100</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>$40</td>
<td>$40</td>
<td>$100</td>
<td>$100</td>
<td>$100</td>
</tr>
<tr>
<td>3</td>
<td>$40</td>
<td>$50</td>
<td>$140</td>
<td>$140</td>
<td>$140</td>
</tr>
<tr>
<td>4</td>
<td>$40</td>
<td>$50</td>
<td>$150</td>
<td>$150</td>
<td>$150</td>
</tr>
<tr>
<td>5</td>
<td>$40</td>
<td>$90</td>
<td>$150</td>
<td>$150</td>
<td>$150</td>
</tr>
<tr>
<td>6</td>
<td>$40</td>
<td>$90</td>
<td>$190</td>
<td>$195</td>
<td>$195</td>
</tr>
<tr>
<td>7</td>
<td>$40</td>
<td>$90</td>
<td>$190</td>
<td>$195</td>
<td>$195</td>
</tr>
<tr>
<td>8</td>
<td>$40</td>
<td>$90</td>
<td>$190</td>
<td>$235</td>
<td>$235</td>
</tr>
<tr>
<td>9</td>
<td>$40</td>
<td>$90</td>
<td>$190</td>
<td>$245</td>
<td>$245</td>
</tr>
<tr>
<td>10</td>
<td>$40</td>
<td>$90</td>
<td>$190</td>
<td>$245</td>
<td>$245</td>
</tr>
</tbody>
</table>

Integer Linear Programming

**Types of Integer Linear Programming Models**

- Graphical Solution for an All-Integer LP
- Spreadsheet Solution for an All-Integer LP
- Application Involving 0-1 Variables
- Special 0-1 Constraints
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: 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:

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

1. \( x_1 \quad 0 \quad 0 \quad 0 \)
2. \( 1 \quad 1 \quad 0 \quad 3 \)
3. \( 2 \quad 0 \quad 6 \quad 0 \)
4. \( 3 \quad 0 \quad 9 \quad 0 \quad \text{optimal solution} \)
5. \( 0 \quad 1 \quad 2 \quad 0 \)
6. \( 1 \quad 1 \quad 5 \quad 0 \)
7. \( 2 \quad 1 \quad 8 \quad 0 \)
8. \( 1 \quad 2 \quad 7 \quad 0 \)

Special 0-1 Constraints

When \( x_i \) and \( x_j \) represent binary variables designating whether projects \( i \) and \( j \) have been completed, the following special constraints may be formulated:

- At most \( k \) out of \( n \) projects will be completed:
  \[ \sum_{i=1}^{n} x_i \leq k \]
- Project \( j \) is conditional on project \( i \):
  \[ x_j \leq x_i \]
- Project \( i \) is a co-requisite for project \( j \):
  \[ x_j \leq x_i \]
- Projects \( i \) and \( j \) are mutually exclusive:
  \[ x_i + x_j \leq 1 \]

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 \( \leq \) 0-1 PROGRAMMING.

Pf.

Scheduling With Release Times

SCHEDULE-RELEASE-TIMES. Given a set of \( n \) jobs with processing time \( t_i \), release time \( r_i \), and deadline \( d_i \), is it possible to schedule all jobs on a single machine such that job \( i \) is processed with a contiguous slot of \( t_i \) time units in the interval \([r_i, d_i]\)?

Claim. SUBSET-SUM \( \leq \) SCHEDULE-RELEASE-TIMES.

Pf. Given an instance of SUBSET-SUM \( w_1, \ldots, w_n \) and target \( W \),

- Create \( n \) jobs with processing time \( t_i = w_i \), release time \( r_i = 0 \), and no deadline \( d_i = 1 + \sum w_j \).
- Create job 0 with \( t_0 = 1 \), release time \( r_0 = W \), and deadline \( d_0 = W+1 \).

Can schedule jobs 1 to \( n \) anywhere but \([W, W+1]\)
Dick Karp (1972)
1985 Turing Award

Polynomial-Time Reductions

3-SAT

INDEPENDENT SET, DIS-HAM-CYCLE, GRAPH 3-COLOR, SUBSET-SUM

VERTEX COVER, HAM-CYCLE, PLANAR 3-COLOR, SCHEDULING

SET COVER, TSP

packing and covering, sequencing, partitioning, numerical