DMS 639

Transport & Telecom

- Underlying mathematical models for both are same — Networks / Graphs
- Issues :
  - Creation of optimal infrastructures to handle the anticipated flow at minimum cost
  - Utilize the current infrastructure to meet demand efficiently.

Course Outline

- Spanning Tree Models
  - Steiner Tree Problem
- Shortest Path Problem
- Network Flow Problem
  - Maximum Flow
  - Minimum Cost Flow
- Fixed Charge Problem
  - Plant Location Problem
  - Fixed Charge Transportation Problem
- Packing Problem
  - Knapsack Problem
  - Bin Packing Problem
- Traveling Salesman Problem
- Vehicle Routing Problem
  ~ TSP + BPP
- Telecom Network Planning

- Chinese Postman Problem
- Large Scale Optimization – Column Generation

Operations Research

- Find the most optimal solution possible
- "Science of Better"
- Optimization Problem:
  Objective Function + Constraints
- Linear Programming:
  Everything is linear
- Steps in Formulation:
  - Identify Decision Variables
  - Identify Objective Function
  - Identify Constraints → Non-negativity constraints too if relevant, along with functional constraints
  - Must be linear
  - Must not miss any constraint(s)

Definitions

- Solution: Any point satisfying the constraints
- Feasible Solution: A solution satisfying non-negativity
- Solution Space / Feasible Region:
  Set of all feasible region
- Optimal Solution: Feasible solution yielding the best value of objective function
- Convex Set: Given two points in the set, the line segment joining them lies in the set.
- Extreme Point: If it is not possible to find two points in the region

s.t. the point lies on the line segment joining the two.

SIMPLEX

- Convert all linear inequalities to equalities:
  - $x_1 \geq 2 \rightsquigarrow x_1 - e_1 = 2$
    $e_1 \geq 0$: Excess/ Surplus Variable
  - $x_1 \leq 2 \rightsquigarrow x_1 + s_1 = 2$
    $s_1 \geq 0$: Slack Variable
- Basic Solution
  - Consider a system of m equations with n variables ($n > m$)
  - Set $(n-m)$ variables to any arbitrary value (say zero), leaving m variables to be solved in m equations, yielding a unique solutions
  - The $(n-m)$ variables are called non-basic variables and the other (m) variables are called basic variables.
  - The columns in the constraint matrix corresponding to the basic variables are linearly independent (LI).
  - The values of the variables are together known as a basic solution.
    A basic solution satisfying non-negativity constraints is called a basic feasible solution (BFS).
  - Degenerate BFS: A BFS with at least one of the basic variables set to zero.
    Otherwise it is a non-degenerate BFS.

LP with Vector Notation

$$\max \quad [c_1 \; c_2 \; \dots \; c_n]_{1\times n} \, [x_1 \; x_2 \; \dots \; x_n]^T_{1\times n}$$

$$\text{s.t.} \quad \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & \dots & \dots & a_{mn} \end{bmatrix}_{m\times n} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}_{n\times 1} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}_{m\times 1},$$

$$[x_1 \; x_2 \; \dots \; x_n]^T \geq [0 \; 0 \; \dots \; 0]^T$$

OR

$$\max \quad c^T x \quad \text{s.t.} \quad AX = b, \quad X \geq 0_n$$

- $X_B = B^{-1} b$ is called the basic solution of $AX = b$; B is called the basis.

  As the choice of basic variables changes, so does B, and so does X_B.

* For LP's, a local optima is also the global optima (thanks to monotonicity of the objective function and the convexity of the feasible region)

Ex: $\max \quad x_1 + 2x_2 \quad \text{s.t.} \quad x_2 \leq 4, \quad x_1 + x_2 \leq 8, \quad 2x_1 - x_2 \leq 10, \quad x_1 \geq 0, x_2 \geq 0$

$$\Rightarrow \quad \begin{aligned} Z - x_1 - 2x_2 &= 0 \\ x_2 + s_1 &= 4 \\ x_1 + x_2 + s_2 &= 8 \\ 2x_1 - x_2 + s_3 &= 10 \end{aligned}$$

$x_1 = 0, \; x_2 = 0$

Simplex Table

C_B	Var in B	Val (X_B)	(1) Y_{x_1}	(2) Y_{x_2}	(0) Y_{s_1}	(0) Y_{s_2}	(0) Y_{s_3}
0	s_1	4	0	1	1	0	0
0	s_2	8	1	1	0	1	0
0	s_3	10	2	-1	0	0	1

Fundamental Theorem

- Every LPP is either feasible, unbounded or infeasible.
- If an LPP has an optimal solution, then it has a basic feasible solution.

& What if LP is not in standard form? How to find a BFS when origin is not in the feasible region?

↳ Use Artificial Variables and then use Big-M method or Two Phase method to eliminate them.

- If the artificial variable is positive in the optimal solution, then the original problem is infeasible.
- In Two-Phase, we can know by the end of Phase I if it is infeasible.

Better use it over Big-M

& Unrestricted Variables

↳ Can take value in $(-\infty, \infty)$, and not restricted to $[0, \infty)$

- Non-positive Variables

$x_j \leq 0 \rightsquigarrow x_j := -y_j$

- Unrestricted Variables

$x_j \in (-\infty, \infty) \rightsquigarrow x_j := y_j - z_j$

Sensitivity Analysis

* Reduced Costs : $z_j - c_j$ for dec. var.
* Dual prices : $z_j - c_j$ for slack variables.
  ↳ Property of a constraint
  - Automatically adapts to alternate solutions.

Cutting Stock Problem

- Woodco sells 3-ft (w_i), 5-ft & 9-ft pieces of lumber
- Woodco's customers demand 25 3-ft boards, 20 (d_i) 5-ft boards & 15 9-ft boards
- Woodco, who must meet its demands by cutting up 17-ft (W) boards, wants to minimize the waste incurred.
- Formulate an LP to help Woodco accomplish its goal.

* Lower bound = $\lceil (\sum w_i d_i)/W \rceil$
  Upper bound = $\sum d_i$
- Can keep decision variables corresponding to each pattern that can be cut
  Ex:

  But, how to enumerate these patterns which can be Θ(Integer Factoring)?

- Some notation:
  W - Length of Stick

I - Set of items
w_i - Length of item
d_i - Demand of item
P - Set of Patterns
x_p - No. of times pattern p is cut
a_{ip} - No. of times item i is cut in pattern $p \in P$

- Objective Function:

Total Length = Demand of Length + Waste

$$\Rightarrow W \sum_p x_p = \sum_i d_i w_i + \text{Waste}$$

$$\Rightarrow \min \text{Waste} \equiv \min W \sum_p x_p - \sum_i d_i w_i$$
$$\equiv \min \sum_p x_p$$

- Constraints:

$$\sum_p a_{ip} x_p \geq d_i \quad \forall \quad i \in I$$

Assumed implicitly that all patterns are valid w.r.t. W

- Finally:

$$\min \sum_p x_p \quad \text{s.t.} \quad \sum_p a_{ip} x_p \geq d_i \quad \forall i \in I,$$
$$0 \leq \underbrace{x_p \in \mathbb{Z}}_{\text{Can be relaxed} \rightarrow \text{just round up}} \quad \forall p \in P$$

- Can find solution using $x_B = B^{-1} b$, $z_j - c_j = C_B B^{-1} a_j - c_j$

- Ex: Patterns: $(5, 0, 0)$, $(0, 3, 0)$, $(0, 0, 1)$

$$\min. \sum x_p$$
$$\text{s.t.} \quad 5x_1 \geq 25, \quad 3x_2 \geq 20, \quad x_3 \geq 15$$

$$B = [x_1 \quad x_2 \quad x_3] = \begin{bmatrix} 5 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \rightarrow B^{-1} = \begin{bmatrix} 1/5 & 0 & 0 \\ 0 & 1/3 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$b = [25 \quad 20 \quad 15]^T$,

$X_B = B^{-1} b = [5 \quad 20/3 \quad 15] \rightarrow 26.7 \rightarrow 27$ (Round-Up Individually)

↳ Not the global optima

⇒ can now use $z_j - c_j$ to find other 'good' patterns

$$\max z_j - c_j$$

$$\equiv C_B B^{-1} a_j - c_j \equiv [1 \; 1 \; 1] B^{-1} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} - 1$$

s.t. $3a_1 + 5a_2 + 9a_3 \leq 17, \quad 0 \leq a_i \in \mathbb{Z}$ (cannot be relaxed)

↳ Sol^n is 1, 1, 1

$Y_{x_4} = B^{-1} [1, 1, 1]^T = [1/5 \; 1/3 \; 1]^T$

Ex:

C_B	B	X_B	Y_{x_1}	Y_{x_2}	Y_{x_3}	Y_{x_4}
1	x_1	5	5	0	0	? ⇒ 1/5
1	x_2	20/3	0	3	0	? ⇒ 1/3
1	x_3	15	0	0	1	? ⇒ 1

↳ Exiting variable

$[1 \; 1 \; 1]$ eliminates $[0 \; 0 \; 1]$

{Intuitive as well ∵ $[0 \; 0 \; 1]$ wastes a lot}

↳
- Update Simplex Table
- Update B^{-1}
- Find solution
- If $z_j - c_j > 0$ ($z_j - c_j$ will ↓ in magnitude w/ each iteration), then pattern is better, repeat the process.
- If $z_j - c_j \leq 0$, stop ∵ solution at hand is optimal.
- Can also stop early if we have already reached the Lower Bound.

(can happen because of IPP vs LPP)

- Takeaway: # of constraints << # of decision variables
- This approach is called **Column Generation**.
  - ↳ Generating Columns on the fly: we do not know them a priori

Seven Bridges of Königsberg

Is there a way to tour the city such that each bridge is used exactly once?

The problem can be reduced to that of graphs.

Graph Theory

- A pair $G = (V, E)$ of sets satisfying $E \subseteq V \times V$
  - V: Set of nodes / points / vertices
  - E: Set of arcs (directed) / lines / edges (undirected)
  - DIGRAPH
    - ↳ Directed Graph
  - Simple Graph
    - ↳ No multiedges
    - ↳ No self-loops
  - Eg: Employee chart, Family ancestry, Road map, Production processes, etc.

Tree and Spanning Tree

- A Tree is an acyclic graph.
- A Spanning Tree is a connected tree.

Muddy City Problem

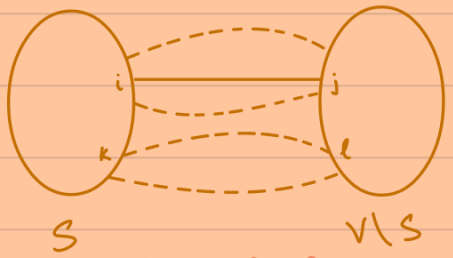
- All roads destroyed due to rain.
- Need to pave some roads so that any house can be reached by any other such that only roads are used.
- Can be modelled using MST's.

Minimum Spanning Tree

- Capacity not an issue
- Total distance not an issue
- Minimal connectivity at minimum cost.
- Eg: Image segmentation, Last-mile connectivity, Reducing data-storage, Clustering, etc.

Optimality Conditions

1) Cut Optimality

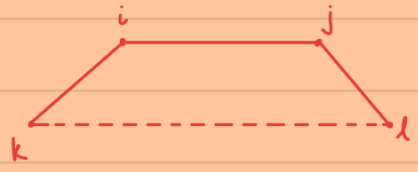


$$c_{ij} \leq c_{kl}$$

c_{ij}: Included in MST

c_{kl}: Not included

2) Path Optimality



$$c_{ij} \leq c_{kl}$$

c_{ij}: Included in MST and path from k to l

c_{kl}: Not included

Algorithms

1) Kruskal's Algorithm

- Based on path optimality.
- Sort edges, select $n-1$ of them whilst ensuring there are no cycles.

2) Prim's Algorithm
- Based on cut optimality.
- Builds the MST incrementally, initially starting from a single node.
- "Do not have to check for cycles"
- Notion of "labels" for each node.

3) Boruvka's (Sollin's) Algorithm
- Hybrid of Prim's & Kruskal's algorithm
- Two steps — "Nearest Neighbour" and "Merge".
- At most $\log_2 n$ iterations of the main loop.

LP formulation of MST

- Dec. Var: x_{ij} : 1 if $e(i,j)$ is picked
  0 if ——— not ———
- Obj. Fun: $\min \sum_{i,j} c_{ij} x_{ij}$
- Constraints:

$$\sum_{i \in S} \sum_{j \in \bar{S}} x_{ij} \geq 1 \quad \forall \; S \subset V, \; \bar{S} := V \setminus S$$

$$x_{ij} \in \{0, 1\}$$

Exponential number of constraints
- Can instead relax the LP
- Solve LP-relax
- Detect cycles

- Add constraints corresponding to that cycle
- Repeat

* Bottleneck Spanning Tree
  Maximum edge's capacity is as low as possible across all trees.
  - Build an MST
  - Find the most expensive edge in the MST
  - Remove all edges costlier than that edge in the MST from the graph.
  - Any Spanning Tree of the reduced graph is a ∴ BST ∴

* Most Uniform Spanning Tree
  Difference b/w largest and smallest edge is minimal

* Maximum / Minimum Degree Spanning Tree
  Degree $\geq k$ / $\leq k$
  - NP - complete

* Isomorphic Spanning Tree
  Given a graph G and a tree T, does ∃ a spanning tree of G isomorphic to T?
  - NP complete

* Bounded Diameter Spanning Tree
  - NP complete

* Leaves Spanning Tree
  $\exists$ ST wt leaves $\geq k$?

* Minimal Cost Reliability Ratio ST
  $w: E \to N$, $r: E \to [0, 1]$
  $w(T) = \sum_{e \in T} w(e)$, $r(T) = \prod_{e \in T} r(e)$
  T s.t. $w(T)/r(T)$ is minimal

Steiner Trees

- A plane tree, with n vertices, along with some other so-called Steiner points
- The Steiner points do not have to be included in the MST but can be used if it reduces the cost of the MST.
- Steiner Ratio

$$\rho = \frac{\text{Total length of M Steiner T}}{\text{M Spanning T}}$$

- Theorem: $\rho \geq \sqrt{3}/2 = 86\%$
  → Optimal

- Theorem: If the edge weights satisfy the triangle inequality, then in the optimal solution, each Steiner point is connected to three or more nodes.

- Lemma: If the arc weights satisfy the triangle inequality, and there are n points to be spanned, the optimal Steiner tree does not contain more than $n-2$ Steiner points.

  Proof. n: No. of regular nodes
  p: No. of Steiner nodes used
  x: Avg. degree of Steiner nodes used
  y: Avg. degree of regular nodes

  Obs: $x \geq 3, \quad y \geq 1$

  Σ degrees $= p \cdot x + n \cdot y = 2 * (p + n - 1) = 2 * \#\text{edges}$

  $\Rightarrow 2*(p+n-1) = px + ny \geq 3p + n$

  $\Rightarrow 2p + 2n - 2 \geq 3p + n$

  $\Rightarrow n - 2 \geq p$ □

Algorithm

S1: Check if $\triangle$ inequality is satisfied. If it is not, replace arc weights with shortest path's distance in the graph.

S2: For each subset of Steiner nodes of size $\leq n-2$, compute the MST with original nodes.

S3: Report the minimum weight amongst the previous MST's.

- The Steiner Tree problem is NP-complete and we have only exponential solutions.

Shortest Path Problem

Given a directed graph $G = (N, A)$, and weights w_{ij} associated with each arc (i, j). find the path of minimum weight between a given pair of nodes.

- Applications:
  - Routing of vehicles
  - Routing of traffic in a telephone network
  - Equipment leasing
  - Approximating piece-wise linear functions
  - System of Difference Constraints
  - Telephone Operator Scheduling
  - Allocating inspection effort of a production line

Equipment Leasing Problem

- An equipment can be leased for some days at the following rate:

Days	Cost
1	1500
3	4000
7	7500
15	14000

- What is the best combination of leases if it is needed for 23 days?

Formulation:

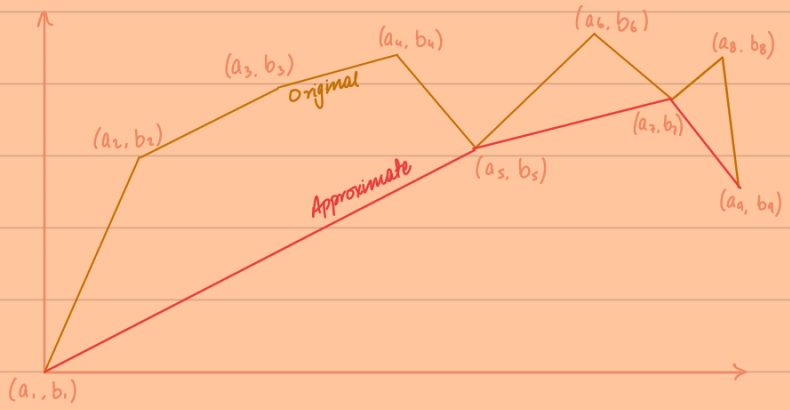
Nodes: Days $\{0, 1, 2, 3, \ldots, 23\}$

Edges: $w_{i,i+1} = 1500$, $w_{i,i+3} = 4000$, $w_{i,i+7} = 7500$, $w_{i,i+15} = 14000$

Starting Node: (0) Ending Node: (23)

Approximating Piecewise Linear Functions

- Accuracy vs Memory trade-off (of approx. vs # of points stored)
- Error $= \sum_i (f_O(x_i) - f_A(x_i))^2$



Formulation:

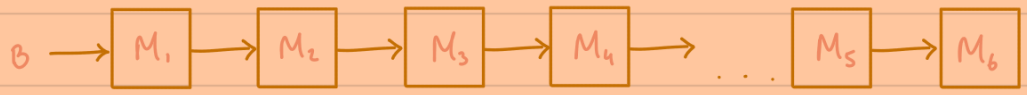
Nodes: Each point where the function breaks linearity

Weights: $W_{ij} = \sum_{i+1}^{j-1} (f_O(x_k) - f_A(x_k))^2$

Starting Node: (a_1, b_1)

Ending Node: (a_n, b_n)

Allocating inspection effort on a production line



α_i : Probability of defect at M_i

p_i : Variable cost of production at M_i

f_{ij} : Fixed cost of inspection at M_j given the last inspection was at M_i

c_{ij} : Variable cost ——

Formulation:

Nodes: Each inspection point

W_{ij} := Prod. Cost + Fixed Cost

$= (B_i \cdot \sum_{k=i}^{k=j} p_i) + (f_{ij} + B_i \cdot c_{ij})$

$$B_i := B \prod_{k=1}^{k=i} (1-\alpha_i)$$

Works because of the way we shall infer the allocation from the solution of the corresponding graph problem

LP Formulation

Decision Variables:

x_{ij} : If the arc (i, j) is chosen in the shortest path from s to t. then 1. otherwise 0

Objective Function:

$$\min. \sum_{i,j} c_{ij} x_{ij}$$

Constraints:

$$\left\{ 1 = \sum_i x_{si}, \quad \sum_i x_{ik} = \sum_j x_{kj}, \quad \sum_j x_{jt} = 1 \right\}$$

$\forall\ k \in V \setminus \{s,t\}$

→ For s-t shortest path.

For shortest path from s to every other node,

$$\sum_i x_{ik} - \sum_j x_{kj} = \begin{cases} -(n-1), & k = s \\ 1, & k \neq t \end{cases}$$

can be thought of as minimizing cost of flow from s

- The shortest path follows the optimal subpath property.

Dijkstra's Algorithm

- SSSP with non-negative edges
- Labels: $u_i - (d_i, p_i)$

d_i: current dist. to i from s

p_i: last node used in the shortest path from s to i

Bellman Ford Algorithm

- Works even for negative-weight edges
- Labels: $u_i^m - (d_i^m, p_i^m)$

d_i^m: Shortest distance to i from s using atmost m hops

Dijkstra vs Bellman-Ford

- Bellman Ford is more flexible (allows negative-weights and hop constraints).
- $O(n^2)$ vs $O(n^3)$
- Label Setting Algo vs Label Correcting

All Pair Shortest Path

- Can run Dijkstra or Bellman-Ford n times but complexity is $O(n^3)$ or $O(n^4)$
  Instead, we can use Floyd-Warshall Algorithm, even with arbitrary weight edges, in $O(n^3)$
- Notation:
  - Each node has a unique label from $[1, n]$ WLOG.
  - $d_{ij}^{(k)}$ - Length of the shortest path from i to j s.t. all the intermediate vertices are from $\{1, \dots, k\}$
- Remarks:
  - In the shortest path, no vertex is repeated
    - Either the cycle is negative
    - If it is positive, then better to avoid the cycle.
  - Either k is not used in the path for $d_{ij}^{(k)}$ $(= d_{ij}^{(k-1)})$ or it is used $(= d_{ik}^{(k-1)} + d_{kj}^{(k-1)})$.
    - $\therefore d_{ij}^{(k)} = \min\left(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right)$

Network Flow Problem

- Flow originates at Source Node, s and terminates at Sink Node, t.
- Each arc (i, j) has a capacity u_{ij} and a cost per unit flow c_{ij}
- Two variants:
  - Maximum Flow Problem → What is the maximum flow?
  - Minimum Cost Flow Problem → Minimum cost to achieve a given flow?

LP formulation

x_{ij} : flow on arc (i, j)

$$\max \sum_{(i,j) \in A} x_{ij}$$

$$\text{s.t.} \quad x_{ij} \leq u_{ij} \quad \forall \ (i, j) \in A,$$

$$\sum_i x_{ik} = \sum_j x_{kj} \quad \forall \ k \in V \setminus \{s, t\}$$

* Some problems admit Integer solutions even when solved as an LP.

- Max-Flow
- Min-Cost Flow
- Transportation Problem
- Assignment Problem

This happens because all the corner points of the feasible region are integral.

↳ $\min \ cx \quad \text{s.t.} \quad Ax \geq b$

Totally uni-modular ← A, integral b

$\Rightarrow x_B = B^{-1} b$ is also integral

- Integral flow, Max-flow Min-cut theorem.

Minimum Cost Flow

- Given a flow and cost of using an edge and its capacity, what is the minimum cost to achieve that flow?

LP formulation

x_{ij} : flow on edge i, j

$$\min. \sum_{i,j} \underbrace{c_{ij}}_{\text{cost per unit flow}} x_{ij}$$

$$\text{s.t.} \quad x_{ij} \leq \underbrace{u_{ij}}_{\text{capacity of the edge}} \quad \forall \ (i,j) \in A$$

$$\sum_{i:(i,k)\in A} x_{ik} - \sum_{j:(j,k)\in A} x_{kj} = \begin{cases} v, & k = t \\ -v, & k = s \\ 0, & k \in N \setminus \{s,t\} \end{cases}$$

M1) Solve using any flow algorithm, ignoring the costs at first.
Then, if there exists an augmenting cycle of negative weight, keep using it (decreases the overall cost).
The optimal flow allocation does not have any negative augmenting cycle.

M2) Solve incrementally using Shortest-Path algorithms until required flow is reached.

Integer Programming

One or more variables required to be an integer.

- Different flavours:
  - Pure Integer Program
  - 0-1 pure IP / binary program
  - Mixed Integer Program
  - Integer Feasibility Problem
  - 0-1 Integer Feasibility Problem
- Caveat: In LPP, variables only denote quantity. But, in IPP, they could mean logical decisions, categories, etc.

- Some standard problems:
  - Knapsack Problem
  - Bin packing problem
  - Plant location problem
  - Either-Or Constraint
    - Capital Budgeting Problem
  - Sudoku

* No efficient solutions for IP's even though IP's have a finite number of feasible solutions (contrast this with LP's that have infinite number of feasible solutions)
  — Optimal points might not be on the boundary of the feasible set (read LP polyhedron).

Branch and Bound

0. Initialization
1. Node Selection
2. Branching
3. Incumbent Updation
4. Fathoming
5. Repeat from (1) on dangling nodes

Some heuristics:

- Initial incumbent solution
  → Better / Faster pruning
- Branching rule
  → Shallower depths
- Node Selection
  → Find Incumbent solution faster

- B&B is an exact algorithm
  Other types: heuristic, approximate

- Cutting plane — cuts the LP feasible region but not the IP feasible region.

Knapsack Problem

- Given items of different weights and values, pack them in a bag of given weight while maximizing the value.
- $\max \sum_i v_i x_i \quad \text{s.t.} \quad \sum_i a_i x_i \leq b, \quad x_i \in \{0, 1\}$

- Upper bound: Greedy approach with fractional items
- Lower bound: Use greedy approach as a heuristic to pick items until capacity is filled.

LP relaxation of the problem
↳ Can be solved w/o simplex → Be greedy

Bin Packing Problem

Minimum number of bins required to pack all the given items.

- $\min \sum_j y_j \quad \text{s.t.} \quad \sum_i a_i x_{ij} \leq W y_j, \; x_{ij}, y_j \in \{0, 1\}$

- Travelling Salesman Problem
  - Assignment Problem
- Vehicle Routing Problem
- Chinese Postman Problem
  - Euler Circuits & Tours
- Graph Coloring
  - Clique vs Independent Set
- Facility Location Problem