12. Linear Programming Applications

- Diet problem
- History
- Network flow
- Branch and bound

Next up: LP geometry, solvers, duality

A linear program

Given $A \in \mathbf{R}^{m \times n}$, $b \in \mathbf{R}^m$, $c \in \mathbf{R}^n$, solve

$$\begin{array}{ll} \underset{x}{\text{minimize}} & c^T x \\ \text{subject to} & Ax = b \\ & x \geq 0 \end{array}$$

- Other variations exist, but all equivalent after reformulations
- Historical importance
- Good solvers (simplex method, interior point methods)
- Generalized to "linear cone" solvers
  - $x \geq 0$ is replaced by x in second-order cone or semidefinite cone
  - Now we can solve lots of convex problems

Diet problem

$$\begin{array}{ll}\underset{x}{\text{minimize}} & c^T x \\ \text{subject to} & Ax = b \\ & x \geq 0\end{array}$$

- I want to lose as much weight as possible
- x_i represents how many servings of food group i to eat
- c_i gives # calories of 1 serving of food from group i
- $a_i^T x = b_i$ encodes nutritional recommendations
- $x \geq 0$ since I can't eat negative food

History

Important fields

- Operations research
  - Started with post-WWII military research
  - Now expands to management sciences
  - Often appears as linear relaxations of important combinatorial problems
  - e.g. assigning people to tasks, routing supplies, strategic planning,...
- Economics
  - 1939: Planning a country's economy (Kantorivich in USSR, Koopmans in US)
  - Planning in business (maximize utility subj. to. resource constraints)
- Combinatorial optimization
  - Linear relaxation gives lower bounds
  - Often used in branch-and-bound solvers

Assignment

Task: assign n people to n tasks

$$\begin{array}{ll} \underset{X \in \mathbf{R}^{n \times n}}{\text{maximize}} & \sum_{ij} X_{ij} W_{ij} \\ \text{subject to} & X^T e = e, \quad Xe = e \\ & X_{i,j} \in \{0, 1\} \end{array}$$

- $X_{ij} = 1 \iff$ person i assigned to task j
- W_{ij} encodes preference of this assignment
- Linear equality constraint ensures only 1 assignment per person and per task
- Combinatorial constraint $X_{i,j} \in \{0, 1\}$ makes problem hard to solve
- Linear relaxation: replace

$$X_{i,j} \in \{0, 1\} \to 0 \leq X_{i,j} \leq 1$$

Routing (aka Traveling Salesman problem)

Task: Assign a supply route for a truck, with n stops

$$\begin{array}{ll}\underset{X \in \mathbf{R}^{n\times n}}{\text{minimize}} & \sum_{ij} X_{ij} W_{ij} \\ \text{subject to} & X^T e = e, \quad Xe = e \\ & \sum_j X_{1,j} = \sum_i X_{i,1} = 1 \\ & \sum_{i \notin S} \sum_{j \in S} X_{ij} \geq 1, \ \forall S \subseteq \{1, ..., n\} \\ & X_{i,j} \in \{0, 1\}\end{array}$$

- $X_{ij} = 1$ if visit i right after j
- Second linear constraint: ensure truck leaves and returns at depo
- Third constraint: ensures route is connected
- Linear relaxation: replace

$$X_{i,j} \in \{0,1\} \to 0 \leq X_{i,j} \leq 1$$

Economics

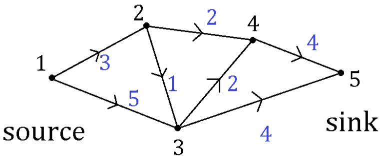
- A gadget is built from two widgets and three fidgets
- Inventory only has 300 widgets
- The fidgets and widgets are stored in boxes, with 3 fidgets and 1 wigdet per box. We need to clear out at least 50 boxes
- How can I maximize the number of gadgets built?
- Problem formulation

$$
\begin{array}{ll}
\underset{x,y}{\text{maximize}} & 2x + 3y \\
\text{subject to} & x < 300 \\
& x + 3y > 50 \\
& x \geq 0,\ y \geq 0 \\
& x, y \quad \text{are integers}
\end{array}
$$

- Linear relaxation: omit last constraint

Network flow

Network flow



$$\begin{bmatrix} 3 & 5 & 0 & 0 & 0 & 0 & 0 \\ -3 & 0 & 1 & 2 & 0 & 0 & 0 \\ 0 & -5 & -1 & 0 & 2 & 0 & 4 \\ 0 & 0 & 0 & -2 & -2 & 4 & 0 \\ 0 & 0 & 0 & 0 & 0 & -4 & -4 \end{bmatrix} \quad \begin{matrix} \text{node} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix}$$

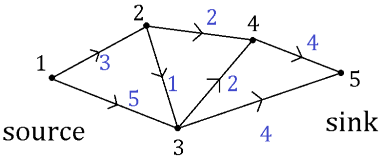
node-arc matrix

Appears in transportation, network routing, planning

- n nodes, m arcs (directed edges)
- $X \in \mathbf{R}^{n \times m}$ is node-arc matrix
- $C_L \leq X \leq C_U$ capacity constraints (width of pipe)
- If no edge between nodes i, j, $(C_L)ij = (C_U)_{ij} = 0$
- Conservation of flow:

$$\sum_j X_{ij} = 0 \quad \text{for all non-source non-sink nodes } i$$

Network flow: Max-flow



$$\begin{bmatrix} 3 & 5 & 0 & 0 & 0 & 0 & 0 \\ -3 & 0 & 1 & 2 & 0 & 0 & 0 \\ 0 & -5 & -1 & 0 & 2 & 0 & 4 \\ 0 & 0 & 0 & -2 & -2 & 4 & 0 \\ 0 & 0 & 0 & 0 & 0 & -4 & -4 \end{bmatrix} \quad \begin{matrix} \text{node} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix}$$

node-arc matrix

$$\begin{array}{lll} \underset{X}{\text{maximize}} & \displaystyle\sum_{i=1}^{n} X_{1,i} & \text{Total flow} \\ \text{subject to} & C_L \leq X \leq C_U & \text{Capacity constraints} \\ & \displaystyle\sum_{j} X_{ij} = 0, \ \forall i \neq 1 & \text{Conservation of flow} \end{array}$$

Branch and bound

Branch and bound

Consider the mixed integer linear program

$$\begin{array}{ll} \underset{x \in \mathbf{R}^n}{\text{minimize}} & c^T x \\ \text{subject to} & Ax = b,\ Cx \le d \\ & x_i \in \{0,1\},\ i = 1, ..., n \end{array}$$

- Generalizes assignment, routing, graph coloring, ...
- Define $p(x) = c^T x$.
- $x \in \mathbf{R}^n$ is **feasible** if

$$Ax = b, \quad Cx \le d, \quad x_i \in \{0,1\},\ i = 1, ..., n$$

Branch and bound

Consider the mixed integer linear program

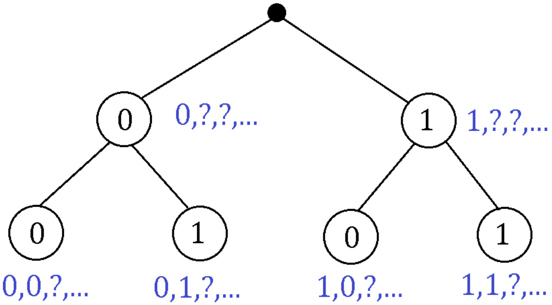
$$\begin{array}{ll}\underset{x\in\mathbf{R}^n}{\text{minimize}} & c^T x \\ \text{subject to} & Ax = b,\ Cx \le d \\ & x_i \in \{0,1\},\ i = 1, ..., n\end{array}$$

- **Upper bound:** For any feasible x, $p(x) \geq p(x^*)$
- **Lower bound:** Consider $\hat{x}$ the solution to **relaxed** problem

$$\begin{array}{ll}\underset{x\in\mathbf{R}^n}{\text{minimize}} & c^T x \\ \text{subject to} & Ax = b,\ Cx \le d \\ & 0 \le x \le e\end{array}$$

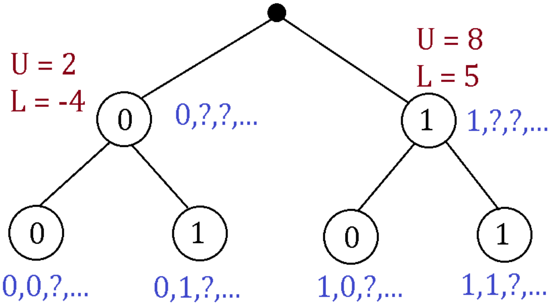
  Then $p(\hat{x}) \leq p(x^*)$

Branch and bound algorithm



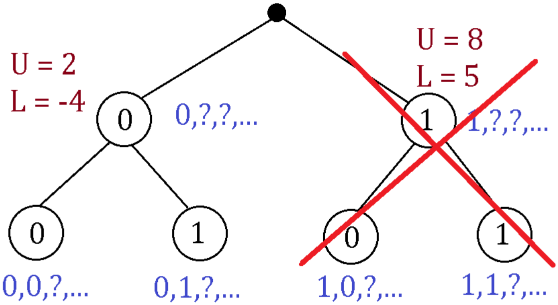
1. Binary tree traverses every possible value of x

Branch and bound algorithm



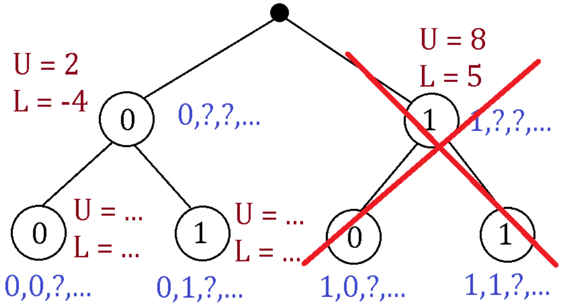
1. Binary tree traverses every possible value of x
2. Breadth-first search: calculate an upper and lower bound given a fixed value

Branch and bound algorithm



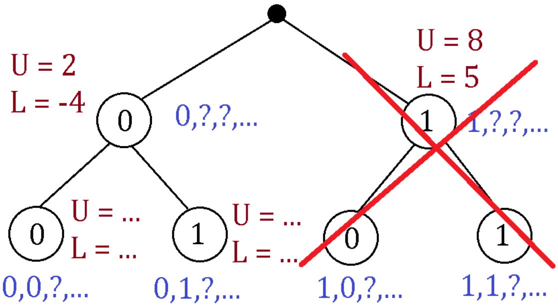
1. Binary tree traverses every possible value of x
2. Breadth-first search: calculate an upper and lower bound given a fixed value
3. If lower bound $>$ upper bound of another node, impossible choice
   - cut node and all descendants

Branch and bound algorithm



1. Binary tree traverses every possible value of x
2. Breadth-first search: calculate an upper and lower bound given a fixed value
3. If lower bound $>$ upper bound of another node, impossible choice
    - cut node and all descendants
4. Continue searching

How to create a good B-B solver



- Better upper bounds (how to guess well?)
- Better lower bounds (Can we do something tighter than LP relaxation?)
- Better ordering of trees

B-B solvers require fast LP solvers, since they may be applied many times!