Linear Programming and Applications

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

Next up: LP geometry, solvers, duality

Linear programming

Given $A \in \mathrm{R}^{m \times n}$, $b \in \mathrm{R}^m$, $c \in \mathrm{R}^n$:

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

- 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

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

- minimum-cost diet
- x_i represents how many servings of food group i to eat
- c_i gives cost of 1 serving of food from group i
- $a_i^T x = b_i$ encodes nutritional recommendations
- $x \geq 0$ since you can't eat negative food

Important fields

- Operations research
  - Started with post-WWII military research
  - many applications in management science
  - often appears as 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 subject 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

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

- $X_{ij} = 1 \iff$ person i assigned to task j
- W_{ij} encodes preference of person i's assignment to task j
- 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
- relaxation: replace binary constraints with interval constraints:

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

Routing (aka, Traveling Salesman problem)

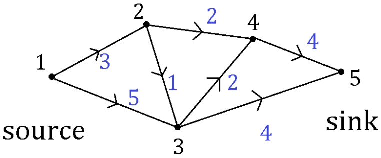
Task: assign a supply route for a truck, with n stops

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

- $X_{ij} = 1$ if visit stop i right after stop j
- second linear constraint: ensure truck leaves and returns at depo ($i = 1$)
- third constraint: ensures route is connected
- relaxation: replace binary constraints with interval constraints:

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

Network flow



$$\Rightarrow \quad \begin{array}{cc} \begin{bmatrix} 3 & 5 & 0 & 0 & 0 & 0 & 0 \\ -3 & 0 & 1 & 2 & 0 & 0 & 0 \\ 0 & -5 & -1 & 0 & 2 & 0 & 4 \\ 0 & 0 & 0 & -2 & -2 & 4 & 0 \\ 0 & 0 & 0 & 0 & 0 & -4 & -4 \end{bmatrix} & \begin{array}{c} \text{node} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \end{array}$$

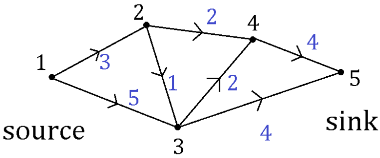
node-arc matrix

Appears in transportation, network routing, planning

- n nodes, m arcs (directed edges)
- $X \in \mathrm{R}^{n \times m}$ records flows from node i through arc j
- $C_L \leq X \leq C_U$ capacity constraints (eg, link capacities)
- if no edge between nodes i and j then $(C_L)_{ij} = (C_U)_{ij} = 0$
- flow conservation:

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

Network flow: Max-flow



$$\begin{array}{c} \\ \left[\begin{array}{ccccccc} 3 & 5 & 0 & 0 & 0 & 0 & 0 \\ -3 & 0 & 1 & 2 & 0 & 0 & 0 \\ 0 & -5 & -1 & 0 & 2 & 0 & 4 \\ 0 & 0 & 0 & -2 & -2 & 4 & 0 \\ 0 & 0 & 0 & 0 & 0 & -4 & -4 \end{array}\right] \end{array} \begin{array}{c} \text{node} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array}$$

node-arc matrix

$\underset{X}{\text{maximize}}$	$\sum_{i=1}^{n} X_{1,i}$	Total flow
subject to	$C_L \leq X \leq C_U$	Capacity constraints
	$\sum_j X_{ij} = 0, \ \forall i \neq 1$	Conservation of flow

Branch and bound

Mixed integer linear program

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

- Generalizes assignment, routing, graph coloring, and more
- $x \in \mathrm{R}^n$ is **feasible** if

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

Branch and bound

Mixed integer linear program

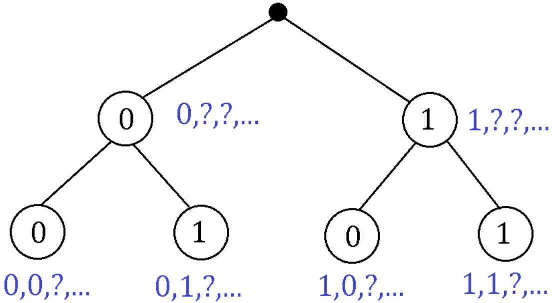
$$\begin{array}{ll} \underset{x \in \mathbf{R}^n}{\text{minimize}} & c^T x \\ \text{subject to} & Ax = b,\ Cx \leq d \\ & x_i \in \{0, 1\},\ i = 1, ..., n \end{array}$$

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

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

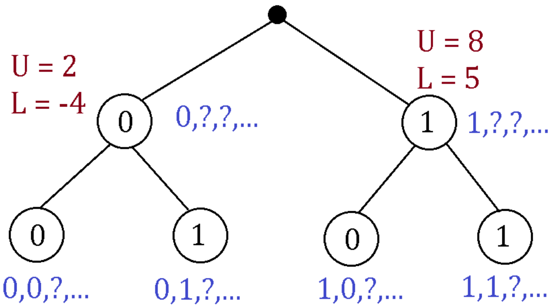
Then $p(\hat{x}) \leq p(x^*)$

Branch and bound algorithm



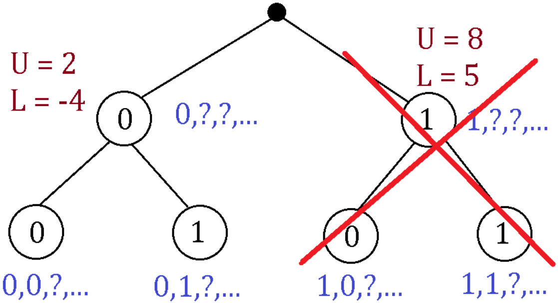
1. Binary tree traverses every possible value of x

Branch and bound algorithm



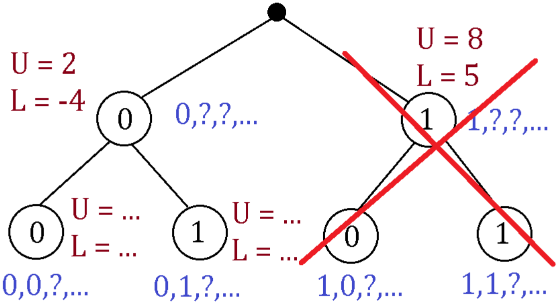
1. Binary tree traverses every possible value of x
2. Breadth-first search: calculate an upper and lower bound given a fixed value

Branch and bound algorithm



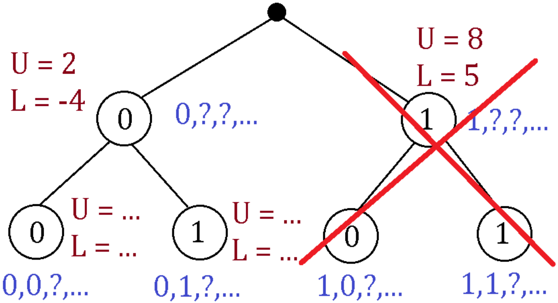
1. Binary tree traverses every possible value of x
2. Breadth-first search: calculate an upper and lower bound given a fixed value
3. If lower bound $>$ upper bound of another node, impossible choice
   - cut node and all descendants

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

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