TODAY: Linear programming
- examples: politics, flow, shortest paths
- general form
- duality
- geometric view
- 2D algorithm
- general algorithms
- low-dimensional algorithms

Politics: how to campaign to win an election
- staff estimates votes obtained per dollar spent advertising in support of a particular issue

\[
\begin{array}{l|ccc}
\text{policy} & \text{urban} & \text{suburban} & \text{rural} \\
\hline
x_1 & \text{building roads} & -2 & 5 & 3 \\
x_2 & \text{gun control} & 8 & 2 & -5 \\
x_3 & \text{farm subsidies} & 0 & 0 & 10 \\
x_4 & \text{gasoline tax} & 10 & 0 & -2 \\
\end{array}
\]

- want to win majority in each demographic population:
  - urban: 100,000
  - suburban: 200,000
  - rural: 50,000

- majority:
  - urban: 50,000
  - suburban: 100,000
  - rural: 25,000

by spending min. amount of money
Algebraic setup:
- let \( x_1, x_2, x_3, x_4 \) denote dollars spent per issue
- goal: minimize \( x_1 + x_2 + x_3 + x_4 \)
  subject to \(-2x_1 + 8x_2 + 0x_3 + 10x_4 \geq 50,000\)
  \(5x_1 + 2x_2 + 0x_3 + 0x_4 \geq 100,000\)
  \(3x_1 - 5x_2 + 10x_3 - 2x_4 \geq 25,000\)
  \(x_1, x_2, x_3, x_4 \geq 0\) (can't unadvertise)

- OPT:
  \[ x_1 = \frac{2,050,000}{111} \approx 18,468 \]
  \[ x_2 = \frac{425,000}{111} \approx 3,829 \]
  \[ x_3 = 0 \]
  \[ x_4 = \frac{625,000}{111} \approx 5,631 \]

  \[ \Rightarrow x_1 + x_2 + x_3 + x_4 = \frac{-3,100,000}{111} \approx 27,928 \]

Linear programming: (LP)
- minimize or maximize linear objective function
- subject to linear inequalities (\& equations)
- variables \( \bar{x} = (x_1, x_2, \ldots, x_d) \)
- objective function: \( \bar{c} \cdot \bar{x} = c_1 x_1 + c_2 x_2 + \cdots + c_d x_d \)
  - can assume maximize; to minimize, \( c \rightarrow -c \)
- inequalities: \( A \bar{x} \leq \bar{b} \) \( \sim \) \( A \) is \( n \times d \)
  - e.g. \( x_1 - x_3 = 7 \) represented as \( (\frac{1}{0} -\frac{1}{0} 0) \bar{x} \leq (7) \)
- thus: \( \max \ \bar{c} \cdot \bar{x} \)
  - s.t. \( A \bar{x} \leq \bar{b} \)
Difference constraints: $x_i - x_j \leq w_{ij}$

- Special case of linear programming where each row of $A$ has one $+1$, one $-1$, & rest $0$s solved by Bellman-Ford

- Indeed, this LP solves $s \rightarrow t$ shortest path:

\[
\begin{align*}
\text{max} \quad & d(t) \\
\text{s.t.} \quad & d(v) - d(u) \leq w(u,v) \quad \text{for each } (u,v) \in E \\
& d(s) = 0 \\
& d(v) \geq 0 \quad \text{for each } v \in V \\
\end{align*}
\]

- No solution $\iff$ neg.-weight cycle reachable from $s$

Maximum flow:

\[
\begin{align*}
\text{max} \quad & \sum_{v \in V} f(s,v) \\
\text{s.t.} \quad & f(u,v) = -f(v,u) \quad \text{for each } u,v \in V \\
& \sum_{u \in V} f(u,v) = 0 \quad \text{for each } v \in V \setminus \{s,t\} \\
& f(u,v) \leq c(u,v) \quad \text{for each } u,v \in V \\
\end{align*}
\]

Minimum cut: $(S, V \setminus S)$

- $S_v = \begin{cases} 1 & \text{if } v \in S \ (0 \text{ or } 1) \\ 0 & \text{otherwise} \end{cases}$

- $x(u,v) = \begin{cases} 1 & \text{if } u \in S \text{ & } v \in V \setminus S \ (0 \text{ or } 1) \\ 0 & \text{otherwise} \end{cases}$

$\Rightarrow \min \sum_{(u,v) \in E} c(u,v) \cdot x(u,v)$

\[
\begin{align*}
\text{s.t.} \quad & S_u - S_v \leq x(u,v) \\
& S_s = 1 \\
& S_t = 0 \\
& x(u,v) \geq 0 \\
\end{align*}
\]

for each $(u,v) \in E$
Integer linear programming: \( \begin{cases} \max \ c \cdot \hat{x} \\ \text{s.t.} \ A \hat{x} \leq b \\ \text{ILP is NP-complete} \end{cases} \)

Duality: \( \max \ c \cdot \hat{x} = \min \ b \cdot \hat{x} \)

\( \begin{aligned} \text{s.t.} & \ A \hat{x} \leq b \\ \hat{x} & \geq 0 \\ \text{s.t.} & \ A^T \tilde{y} \geq \hat{c} \\ \tilde{y} & \geq 0 \end{aligned} \)

"standard form" (any LP can be written this way)

- special case: if LP is unbounded (Opt=±∞)
  then dual LP is infeasible (no solution)

\( \Rightarrow \) max-flow min-cut theorem

Geometric view:
- \( \hat{x} \) is a point in \( \mathbb{R}^d \)
- \( \hat{c} \) is a direction vector (length is irrelevant)
- \( \max \ c \cdot \hat{x} = \) want \( \hat{x} \) most in the direction \( \hat{c} \)
- constraint \( \hat{a} \cdot \hat{x} \leq b \) = halfspace bounded by plane
- together constraints form a polytope
  with \( \leq n \) polygonal facets
- possibly unbounded ("open")
- can rotate entire problem so that goal is to find highest point \( \hat{x} \) in polytope \( \hat{c}=(\frac{e_i}{1}) \)

2D:
- polytope = polygon
- halfspace = halfplane
- plane = line
- irrelevant constraint
Incremental algorithm: for 2D
- maintain polygon for first i-1 constraints
- add i-th constraint:
  - 0, 1, or 2 intersections
- can find in O(lg n) similar to binary search
- discard now-irrelevant constraints
- must maintain polygon in balanced search tree
- O(n lg n) time

Higher dimensions:
\# vertices = (\binom{n}{d}) \approx n^d \quad \text{in the worst case}:

General algorithms:
- simplex algorithm: \( \hat{x} \) walks from vertex to vertex in \( \mathbb{R}^d \)
  \( \sim \) practical but worst-case exponential
- ellipsoid algorithm: guarantee OPT ellipsoid; reduce ellips.
  \( \sim \) first poly time, useful in theory, impractical
- interior-point method: \( \hat{x} \) flows inside polytope vaguely \( \hat{c} \)
  \( \sim \) poly. time \& quite practical
- random sampling: [Bertsimas \& Vempala 2004]
  Sample to estimate center of mass, slice OPT estimate, repeat
- randomized simplex: [Kelner \& Spielman 2006]
  reduce to testing boundedness; randomize \( b \); simplex; repeat
Low-dimensional algorithm:
- given halfspaces $H$ & objective vector $\vec{c}$
- pick any $h \in H$
- recurse $(H \setminus \{h^2\}, \vec{c}) \rightarrow x$
- if $x \in h$: return $x$ \hspace{1cm} (h didn’t affect OPT)
- else: OPT must be on $\pi = \text{plane}(h)$
  recurse $(\{h’ \cap \pi \mid h’ \in H \setminus \{h^2\}\}, \vec{c} \text{ projected on } \pi)$

Time: $T(n,d) = T(n-1,d) + T(n-1,d-1) + O(nd)$
\[ \leq c(n-1)^d + c(n-1)^{d-1} + O(nd) \]
\[ = c(n-1)^{d-1}((n-1)+1) + O(nd) \]
\[ = c n(n-1)^{d-1} + O(nd) \]
\[ \leq c n^d \text{ for } d \geq 2 \text{ & } c \text{ large enough} \]

$T(n,1) \leq c n$ by simple max/min

Seidel’s algorithm: [1991]
- same, but pick $h \in H$ uniformly at random
- $Pr \{\exists x \in L \cap h^2 \text{ necessary to bound } \text{OPT} \} \leq d/n$

$\Rightarrow E[T(n,d)] = E[T(n-1,d)] + \frac{d}{n}[E[T(n-1,d-1)] + O(nd)]$
\[ \leq cd \cdot d!(n-1) + \frac{d}{n}[c(d-1)(d-1)!/(n-1) + O(nd)] \]
\[ = cd \cdot d!(n-1) + c(d-1)d! \cdot \frac{(n-1)}{n} + O(d^2) \]
\[ \leq cd \cdot d! \cdot n - cd! + O(d^2) \]
\[ \leq c \cdot d \cdot d! \cdot n \text{ for large enough } c \]

- linear time for any fixed $d$
- polynomial for $d = O(\lg n / \log \log n)$
Best low-dim algorithm: \(O(d^2n + 2^{O(d \log d)})\)  
[see Gärtner & Welzl 1996]  

OPEN: can LP be solved in \(\text{poly}(n,d)\) time?  
- general algorithms above achieve \(\text{poly}(n,d,b)\)  
  for input & output precision of \(b\) bits