**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

<table>
<thead>
<tr>
<th>demographic</th>
<th>urban</th>
<th>suburban</th>
<th>rural</th>
</tr>
</thead>
<tbody>
<tr>
<td>e.g. policy</td>
<td>urban</td>
<td>suburban</td>
<td>rural</td>
</tr>
<tr>
<td>$x_1$ building roads</td>
<td>-2</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>$x_2$ gun control</td>
<td>8</td>
<td>2</td>
<td>-5</td>
</tr>
<tr>
<td>$x_3$ farm subsidies</td>
<td>0</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>$x_4$ gasoline tax</td>
<td>10</td>
<td>0</td>
<td>-2</td>
</tr>
</tbody>
</table>

- want to win majority in each demographic
- population: 100,000 200,000 50,000
- majority: 50,000 100,000 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
  \[
  \begin{align*}
  -2x_1 + 8x_2 + 0x_3 + 10x_4 & \geq 50,000 \\
  5x_1 + 2x_2 + 0x_3 + 0x_4 & \geq 100,000 \\
  3x_1 - 5x_2 + 10x_3 - 2x_4 & \geq 25,000 \\
  x_1, x_2, x_3, x_4 & \geq 0 \quad \text{(can't unadvertise)}
  \end{align*}
  \]
- OPT:
  \[
  \begin{align*}
  x_1 &= \frac{2,050,000}{111} \approx 18,468 \\
  x_2 &= \frac{4,250,000}{111} \approx 3,829 \\
  x_3 &= 0 \\
  x_4 &= \frac{6,250,000}{111} \approx 5,631
  \end{align*}
  \]
  \[
  \Rightarrow x_1 + x_2 + x_3 + x_4 \approx 31,500 \quad \text{(can't unadvertise)}
  \]

Linear programming: (LP)
- minimize or maximize linear objective function
- subject to linear inequalities (\& equations)
- variables \( \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{pmatrix} \)
- objective function: \( \mathbf{c} \cdot \mathbf{x} = c_1 x_1 + c_2 x_2 + \cdots + c_d x_d \)
  - can assume maximize, to minimize, \( c \to -c \)
- inequalities: \( A \mathbf{x} \leq \mathbf{b} \quad \text{\&} \quad A \mathbf{x} \geq \mathbf{b} \mathbf{c} \cdot \mathbf{x} \)
  - e.g. \( x_1 - x_3 \geq 7 \) represented as \( \begin{pmatrix} 1 & 0 & -1 & 0 & 7 \end{pmatrix} \mathbf{x} \leq (7) \)
- thus: \( \max \mathbf{c} \cdot \mathbf{x} \)
  s.t. \( A \mathbf{x} \leq \mathbf{b} \)
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 0s
- solved by Bellman-Ford
- indeed, this LP solves $s \to t$ shortest path:
  \[
  \begin{align*}
  \text{max} & \quad d[t] \\
  \text{s.t.} & \quad d[v] - d[u] \leq w(u,v) \quad \text{for each } (u,v) \in E \\
  & \hfill d[s] = 0 \\
  & \hfill d[v] \geq 0 \quad \text{for each } v \in V \\
  \end{align*}
  \]
- no solution $\iff$ neg.-weight cycle reachable from $s$

Maximum flow:

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

= $|f|$ skew sym.

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

- $S_v = 1$ if $v \in S$? (0 or 1) $\iff$ not forced $\iff$ automatic
- $x(u,v) = 1$ if $u \in S$ & $v \in V \setminus S$? (0 or 1)
- $\Rightarrow \min \sum_{(u,v) \in E} c(u,v) \cdot x(u,v) $ for each $(uv) \in E$

\[
\begin{align*}
0 \leq & \sum_{u \in V} x(u,v) \\
& \leq 1 \\
0 & \leq S_u \leq 1 \\
0 & \leq S_v \leq 1 \\
S_s & = 1 \\
S_t & = 0 \\
x(u,v) & \geq 0
\end{align*}
\]
Integer linear programming:

\[
\begin{aligned}
\text{max } & \mathbf{c} \cdot \mathbf{x} \\
\text{st. } & A\mathbf{x} \leq \mathbf{b} \\
& x_1, x_2, \ldots, x_i \text{ integral}
\end{aligned}
\]

ILP is NP-complete

Duality:

\[
\begin{aligned}
\text{max } & \mathbf{c} \cdot \mathbf{x} = \min b^* \cdot \mathbf{y} \\
\text{st. } & A\mathbf{x} \leq \mathbf{b} \\
& \mathbf{A}^\top \mathbf{y} \geq \mathbf{c} \\
& \mathbf{y} \geq \mathbf{0}
\end{aligned}
\]

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

- special case: if LP is unbounded \((\text{OPT}=\pm \infty)\), then dual LP is infeasible (no solution)

\[\Rightarrow\text{ max-flow min-cut theorem}\]

Geometric view:

- \(\mathbf{x}\) is a point in \(\mathbb{R}^d\)
- \(\mathbf{c}\) is a direction vector (length is irrelevant)
- \(\text{max } \mathbf{c} \cdot \mathbf{x} = \text{want } \mathbf{x}\) most in the direction \(\mathbf{c}\)
- constraint \(\mathbf{a} \cdot \mathbf{x} \leq b\) = 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 \(\mathbf{x}\) in polytope \((\mathbf{c}=(\frac{8}{3}))\)

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
- \(\Rightarrow\) must maintain polygon in balanced search tree
- \(O(n lg n)\) time

Higher dimensions:
\[
\# \text{ vertices} = \binom{n}{d} \sim n^d \quad \text{in the worst case}: \]

General algorithms:
- simplex algorithm: \(\hat{x}\) walks from vertex to vertex in dim. \(\hat{c}\)
  - practical but worst-case exponential (see CLRS)
- ellipsoid algorithm: guarantee \(OPT\) ellipsoid; reduce ellips.
  - first poly time, useful in theory, impractical
- interior-point method: \(\hat{x}\) flows inside polytope vaguely \(\Rightarrow\hat{c}\)
  - 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 $(H-\Sigma h_3, \vec{c}) \Rightarrow x$
- if $x \in h$: return $x$ $(h$ didn't affect OPT$)$
- else: OPT must be on $\pi = \text{plane}(h)$
  recurse $(\{h' \cap \pi \mid h' \in H-\Sigma h_3, \vec{c} \text{ projected on } \pi\})$

Time: $T(n,d) = T(n-1,d) + T(n-1,d-1) + O(nd)$
  $\leq c(n-1)^d + c(n-1)^{d-1} + O(nd)$ \{inductive guess\}
  $= c n(n-1)^{d-1} + O(nd)$
  $\leq c n^d$ for $d \geq 2$ & $c$ 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
- $\Pr_x [x \in h_3] = \Pr_x [h \text{ necessary to bound } \text{OPT}^3 \leq d/n$
  $\Rightarrow E[T(n,d)] = E[T(n-1,d)] + \frac{d}{n} [E[T(n-1,d-1)] + O(nd)]$
  $\leq c d \cdot d! (n-1) + \frac{d}{n} [c(d-1)(d-1)! (n-1) + O(nd)]$ \{inductive guess\}
  $= c d \cdot d! (n-1) + c(d-1) d! \frac{n-1}{n} + O(d^2)$
  $\leq c d \cdot d! \cdot n - c d! + O(d^2)$
  $\leq c d \cdot d! \cdot n$ for large enough $c$

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

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