Recap

- Network Flow Problems
  - Max-Flow Min Cut Theorem
  - Ford-Fulkerson
  - Augmenting Paths
  - Residual Flow Graph
- Integer Solutions (given integral capacities)
- Capacity Scaling Algorithm
- Dinic’s Algorithm
- Applications of Maximum Flow
- Minimum Bipartite Matching
- Marriage Theorem (Hall/Frobenius)
- Disjoint Paths (Menger’s Theorem)
- Baseball Elimination
- Circulation with Demands
- Many Others...

Linear Programming

- Even more general than Network Flow!
- Many Applications
  - Network Flow Variants
  - Transportation
  - Multi-Commodity Flow Problems
  - Supply-Chain Optimization
  - Operations Research
- Entire Courses Devoted to Linear Programming!
- Our Focus
  - Using Linear Programming as a tool to solve algorithms problems
  - We won’t cover algorithms to solve linear programs in any depth

Motivating Example: Time Allocation

168 Hours in Each Week to Allocate as Follows

<table>
<thead>
<tr>
<th>Activity</th>
<th>Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Studying (S)</td>
<td></td>
</tr>
<tr>
<td>Party (P)</td>
<td></td>
</tr>
<tr>
<td>Other (E)</td>
<td></td>
</tr>
</tbody>
</table>

Constraints:

- [Total Hours] 5 + 6 + 6 = 168
- [Maximum Sanity] 5 + 6 = 17
- [Sane] 6 ≥ 56
- [Party Courses] 2 * 6 = 12 (too little sleep, needs more difficulty to study)

Question 1: Can we satisfy all of the constraints? (Maintain Sanity + Pass Courses)
Answer: Yes. One feasible solution is S=80, P=20, E=68

Credit for Example: Avrim Blum
**Linear Program Definition**

- Variables: \(x_1, \ldots, x_n\)
- No linear inequalities in these variables (constraints are CQ)
- Examples:
  - \(0 \leq x_1 \leq 1\)
  - \(x_1 + x_2 + \cdots + x_n \leq 10\)
- **Linear Objective Function**
- Example:
  - Maximize \(3x_1 + 4x_2\)
  - Minimize \(5x_1 - 2x_2\)
- Goal:
  - Find values for \(x_1, \ldots, x_n\) satisfying all constraints, and
  - Maximize the objective
  - Minimize the objective
- **Non-Linear Constraints**
- No objective function

**Network Flow as a Linear Program**

- **Example**
  - Variables: \(x_1, x_2, x_3, x_4, x_5, x_6\)
- Goal: Maximize \(x_5\)
- Subject to:
  - \(3x_2 + x_3 = 10\)
  - \(x_1 + x_2 + x_4 + x_5 = 3\)
- **Requirements**
  - All constraints are linear inequalities in variables \(S, P, F\)
  - The objective function is linear
- **Example Non-Linear Constraint**
  - \(x_5 + x_6 = 6\)
  - \(x_4 + x_5 = 2\)

---

**Linear Program Example**

- **Goal**: Maximize \(3x_1 + 4x_2\)
- **Subject to**:
  - \([3, 5, 0, 0] \cdot [x_1, x_2, x_3, x_4] \leq [10, 15, 0, 0]\)
  - \([0, 5, 0, 0] \cdot [x_1, x_2, x_3, x_4] \leq 10\)
- **Requirements**
  - All constraints are linear inequalities in variables \(S, P, F\)
  - The objective function is linear
- **Example Non-Linear Constraint**
  - \(x_5 + x_6 = 6\)
  - \(x_4 + x_5 = 2\)

---

**Solving a Linear Program**

- Simplex Algorithm (1940s)
  - Not guaranteed to run in polynomial time
  - We can find bad examples, but...
  - The algorithm is efficient in practice!
- Ellipsoid Algorithm (1980)
  - Polynomial time (huge theoretical breakthrough), but...
  - Slow in practice
- Newer Algorithms
  - Karmarkar’s Algorithm
  - Competitive with Simplex
  - Polynomial Time
Algorithmic Idea: Direction of Goodness

Goal: Maximize $2x_1 + 3x_2$ \(c=(2,3)\)

Linear Programming

Theorem: Maximum value achieved at vertex (extreme point)

Definition: Let \(P\) be the set of all feasible points in a linear program. We say that a point \(p \in P\) is an extreme point (vertex) if every line segment \(l \subseteq P\) that has completely in \(P\) and contains \(p\) has \(p\) as an endpoint.

Linear Programming

Theorem: Maximum value achieved at vertex (extreme point)

Definition: Let \(P\) be the set of all feasible points in a linear program. We say that a point \(p \in P\) is an extreme point (vertex) if every line segment \(l \subseteq P\) that has completely in \(P\) and contains \(p\) has \(p\) as an endpoint.

Linear Programming

Theorem: Maximum value achieved at vertex (extreme point)

Definition: Let \(P\) be the set of all feasible points in a linear program. We say that a point \(p \in P\) is an extreme point (vertex) if every line segment \(l \subseteq P\) that has completely in \(P\) and contains \(p\) has \(p\) as an endpoint.

Linear Programming

Theorem: Maximum value achieved at vertex (extreme point)

Definition: Let \(P\) be the set of all feasible points in a linear program. We say that a point \(p \in P\) is an extreme point (vertex) if every line segment \(l \subseteq P\) that has completely in \(P\) and contains \(p\) has \(p\) as an endpoint.

Observation: Each extreme point lies at the intersection of (at least) two constraints.

Theorem: A vertex is an optimal solution if there is no better neighboring vertex.
Algorithmic Idea: Vertex Walking

Goal: Maximize $2x_1 + 3x_2$ \hspace{1cm} \mathbf{c} = (2, 3)$

---

Ellipsoid Algorithm: Solves Feasibility Problem

Case 1: Center of ellipse is in \( \mathcal{F} \)
Step 1: Find large ellipse containing feasible region

---

Case 2: Center of ellipse not in \( \mathcal{F} \)
Step 1: Find large ellipse containing feasible region
Finding the Optimal Point with Ellipsoid Algorithm

Finding the Optimal Point with Ellipsoid Algorithm

Linear Programming in Practice

Linear Programming in Practice

More Linear Programming Examples

More Linear Programming Examples

More Linear Programming Examples

Solving in Mathematica

Solving in Mathematica

Copyright 2000, Kevin Wayne
2-Player Zero-Sum Games

Example: Rock-Paper-Scissors

<table>
<thead>
<tr>
<th>Alice/Bob</th>
<th>Rock</th>
<th>Paper</th>
<th>Scissors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rock</td>
<td>0</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>Paper</td>
<td>1</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>Scissors</td>
<td>-1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Alice wins → Bob loses (and vice-versa)

Minimax Optimal Strategy (possibly randomized) best strategy you can find given that opponent is rational (and knows your strategy)

Minimax Optimal for Rock-Paper-Scissors: play each action with probability 1/3.

Alice's View of Rewards (Bob's are reversed)

Shooter-Goalie

Example: Shooter-Goalie

<table>
<thead>
<tr>
<th>Outcome</th>
<th>Bob's Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block Left</td>
<td>0.9</td>
</tr>
<tr>
<td>Block Right</td>
<td>1/3</td>
</tr>
</tbody>
</table>

Shooter scores 80% of time when shooter aims right and goalie blocks left

Minimax Optimal Strategy (possibly randomized) best strategy you can find given that opponent is rational (and knows your strategy)

How can we find Minimax Optimal Strategy?

Finding Minimax Optimal Solution using Linear Programming

Extra Slides

Circulation with Demands

Circulation problem: given (V, E, d), does there exist a circulation?
Necessary condition: \[ \sum d(v) = \sum d(v) = D \]

Proof. Sum conservation constraints for every demand node \( v \).

Circulation with Demands and Lower Bounds

Max flow formulation.

- Add new source \( s \) and sink \( t \).
- For each \( v \) with \( d(v) < 0 \), add edge \((s, v)\) with capacity \(-d(v)\).
- For each \( v \) with \( d(v) > 0 \), add edge \((v, t)\) with capacity \(d(v)\).

Claim: \( G \) has circulation iff \( G' \) has max flow of value \( D \).

Integrality theorem. If all capacities and demands are integers, and there exists a circulation, then there exists one that is integer-valued.

Proof. Follows from max flow formulation and integrality theorem for max flow.

Characterization. Given \((V, E, c, d)\), there does not exists a circulation iff there exists a node partition \((A, B)\) such that

\[ \sum \delta_A d(v) + \cap(A, B) \leq 0 \]

Proof idea. Look at min cut in \( G' \).

Circulation problem with lower bounds. Given \((V, E, f, c, d)\), does there exists a circulation?

Idea. Model lower bounds with demands.

- Send \( \ell(e) \) units of flow along edge \( e \).
- Update demands of both endpoints.

Theorem. There exists a circulation in \( G \) iff there exists a circulation in \( G' \). If all demands, capacities, and lower bounds in \( G \) are integers, then there is a circulation in \( G \) that is integer-valued.

Proof sketch. \( f'(x) = f(x) - \ell(x) \) is a circulation in \( G \). If all demands, capacities, and lower bounds in \( G \) are integers, then there is a circulation in \( G \) that is integer-valued.
Survey Design

7.8 Survey Design

Algorithm. Formulate as a circulation problem with lower bounds.
- Include an edge \((i,j)\) if consumer \(i\) owns product \(j\).
- Integer circulation \(\Rightarrow\) feasible survey design.

Goal. Design a survey that meets these specs, if possible.

Bipartite perfect matching. Special case when \(c_i = c_i' = 1\).

Image Segmentation

7.10 Image Segmentation

Foreground / background segmentation.
- Label each pixel in picture as belonging to foreground or background.
- \(V = \text{set of pixels}, E = \text{pairs of neighboring pixels}\).
- \(a_i \geq 0\) is likelihood pixel \(i\) is foreground.
- \(b_i \geq 0\) is likelihood pixel \(i\) is background.
- \(p_{ij} \geq 0\) is separation penalty for labeling one of \(i\) and \(j\) as foreground, and the other as background.

Goals.
- Accuracy: if \(a_i > b_i\) in isolation, prefer to label \(i\) in foreground.
- Smoothness: if many neighbors of \(i\) are labeled foreground, we should be inclined to label \(i\) as foreground.

Find partition \((A,B)\) that maximizes:

\[
\sum_{i \in A} \sum_{j \in B} a_{ij} - \sum_{(i,j) \in E} p_{ij}
\]

foreground background
Image Segmentation

Formulate as min cut problem.
- Maximization.
- No source or sink.
- Undirected graph.

Turn into minimization problem.
- Maximizing $\sum_{i \in A} + \sum_{j \in B} - \sum_{(i, j) \in E}$
  - is equivalent to minimizing
  - or alternatively

Consider min cut $(A, B)$ in $G'$.
- $A = \text{foreground}$.
- Precisely the quantity we want to minimize.

$G' = (V', E')$.
- Add source to correspond to foreground; add sink to correspond to background.
- Use two anti-parallel edges instead of undirected edge.

$\text{cap}(A, B) = \sum_{i \in A} + \sum_{j \in B} + \sum_{(i, j) \in E}$

Project Selection

Projects with prerequisites
- Set $P$ of possible projects. Project $v$ has associated revenue $p_v$.
- Some projects generate money: create interactive e-commerce interface, redesign web page.
- Others cost money: upgrade computers, get site license.
- Set of prerequisites $E$. If $(v, w) \in E$, can’t do $v$ unless also do $w$.
- A subset of projects $A \subseteq P$ is feasible if the prerequisite of every project in $A$ also belongs to $A$.

Project selection. Choose a feasible subset of projects to maximize revenue.

Prerequisite graph.
- Include an edge from $v$ to $w$ if can’t do $v$ without also doing $w$.
- $(v, w, x)$ is feasible subset of projects.
- $(v, x)$ is infeasible subset of projects.
Min cut formulation:
- Assign capacity $\infty$ to all prerequisite edge.
- Add edge $(s, v)$ with capacity $-pv$ if $pv > 0$.
- Add edge $(v, t)$ with capacity $-pv$ if $pv < 0$.
- For notational convenience, define $p_s = p_t = 0$.

Claim. $(A, B)$ is min cut iff $A - \{s\}$ is optimal set of projects.
- Infinite capacity edges ensure $A - \{s\}$ is feasible.
- Max revenue because:
  \[ \text{cap}(A, B) = \sum_{v \in B} pv - \sum_{v \in A} pv \]

Open Pit Mining
- Open-pit mining: (studied since early 1960s)
  - Blocks of earth are extracted from surface to retrieve ore.
  - Each block $v$ has net value $pv = \text{value of ore} - \text{processing cost}$.
  - Can't remove block $v$ before $w$ or $x$.

Dancing problem.
- Exclusive Ivy league party attended by $n$ men and $n$ women.
- Each man knows exactly $k$ women; each woman knows exactly $k$ men.
- Acquaintances are mutual.
- Is it possible to arrange a dance so that each woman dances with a different man that she knows?

Mathematical reformulation: Does every $k$-regular bipartite graph have a perfect matching?
- Ex. Boolean hypercube.

Feasible matrix rounding.
- Given a $p$-by-$q$ matrix $D = (d_{ij})$ of real numbers.
- Row $i$ sum = $a_i$, column $j$ sum = $b_j$.
- Round each $d_{ij}$ up or down to integer so that sum of rounded elements in each row (column) equals row (column) sum.
- Original application: publishing US Census data.

Census Tabulation (Exercise 7.39)
- Goal: Find a feasible rounding, if one exists.

k-Regular Bipartite Graphs Have Perfect Matchings
- Theorem. (König 1916, Frobenius 1917) Every $k$-regular bipartite graph has a perfect matching.
- Pf. Size of max matching = value of max flow in $G'$. Consider flow:
  \[ f(v, w) = \begin{cases} \frac{1}{k} & \text{if } (u, v) \in E \\ 1 & \text{if } u = s \text{ or } v = t \\ 0 & \text{otherwise} \end{cases} \]

Original matrix

<table>
<thead>
<tr>
<th></th>
<th>women</th>
<th>man</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.5</td>
<td>17</td>
<td>3</td>
</tr>
<tr>
<td>6.8</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>7.3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>17.6</td>
<td>10</td>
<td>15</td>
</tr>
</tbody>
</table>

Feasible rounding

<table>
<thead>
<tr>
<th></th>
<th>women</th>
<th>man</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.5</td>
<td>17</td>
<td>3</td>
</tr>
<tr>
<td>6.8</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>7.3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>17</td>
<td>10</td>
<td>15</td>
</tr>
</tbody>
</table>
### Feasible Matrix Rounding

- Given a $p \times q$ matrix $D = \{d_{ij}\}$ of real numbers.
- Row $i$ sum $= a_i$, column $j$ sum $= b_j$.
- Round each $d_{ij}$, $a_i$, $b_j$ up or down to integer so that sum of rounded elements in each row (column) equals row (column) sum.
- Original application: publishing US Census data.

**Goal:** Find a feasible rounding, if one exists.

**Remark:** “Threshold rounding” can fail.

<table>
<thead>
<tr>
<th>Original Matrix</th>
<th>Feasible Rounding</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.35 0.35 0.35 1.05</td>
<td>0 0 1 1</td>
</tr>
<tr>
<td>0.55 0.55 0.55 1.65</td>
<td>1 1 0 2</td>
</tr>
<tr>
<td>0.5 0.5 0.7</td>
<td>1 1 1</td>
</tr>
</tbody>
</table>

---

### Theorem

**Theorem:** Feasible matrix rounding always exists.

**Proof:** Formulate as a circulation problem with lower bounds.
- Original data provides circulation (all demands = 0).
- Integrality theorem $\Rightarrow$ integral solution $\Rightarrow$ feasible rounding.

<table>
<thead>
<tr>
<th>Lower Bound</th>
<th>Upper Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.54 6.8 7.3 17.74</td>
<td></td>
</tr>
<tr>
<td>9.4 4.2 0.7 12.7</td>
<td></td>
</tr>
<tr>
<td>3.6 3.2 6.5 13.2</td>
<td></td>
</tr>
<tr>
<td>16.54 10.4 14.8</td>
<td></td>
</tr>
</tbody>
</table>