Recap

- Network Flow Problems
- Max-Flow Min Cut Theorem
- Ford Fulkerson
  - Augmenting Paths
  - Residual Flow Graph
  - Integral Solutions (given integral capacities)
- Capacity Scaling Algorithm
- Dinic’s Algorithm
- Applications of Maximum Flow
  - Maximum 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
  - Taxation
  - 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

- Studying (S)
- Partying (P)
- Everything Else (E)

Constraints:
- [168 Hours] \( S + P + E = 168 \)
- [Maintain Sanity] \( P + E \geq 70 \)
- [Pass Courses 1] \( S \geq 60 \)
- [Pass Courses 2] \( 2S + E - 3P \geq 150 \) (too little sleep, and/or too much partying makes it more difficult to study)

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

Motivating Example: Time Allocation

168 Hours in Each Week to Allocate as Follows

- Studying (S)
- Partying (P)
- Everything Else (E)

Constraints:
- [168 Hours] \( S + P + E = 168 \)
- [Maintain Sanity] \( P + E \geq 70 \)
- [Survive] \( E \geq 56 \)
- [Pass Courses 1] \( S \geq 60 \)
- [Pass Courses 2] \( 2S + E - 3P \geq 150 \) (too little sleep, and/or too much partying makes it more difficult to study)

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

Motivating Example: Time Allocation

168 Hours in Each Week to Allocate as Follows

- Studying (S)
- Partying (P)
- Everything Else (E)

Constraints:
- [168 Hours] \( S + P + E = 168 \)
- [Maintain Sanity] \( P + E \geq 70 \)
- [Survive] \( E \geq 56 \)
- [Pass Courses 1] \( S \geq 60 \)
- [Pass Courses 2] \( 2S + E - 3P \geq 150 \) (too little sleep, and/or too much partying makes it more difficult to study)

Objective Function: \( 2P + E \) [Maximize Happiness]

Question 2: Can we find a feasible solution which maximizes the objective function?
Linear Program Definition

- Variables: \( x_1, \ldots, x_n \)
- \( m \) linear inequalities in these variables (equalities are OK)
- Examples
  - \( 0 \leq x_1 \leq 1 \)
  - \( x_1 + x_4 + 3 x_{10} - 7 x_{11} \leq 4 \)
  - \( 2 S + E \leq 3 P \geq 150 \)
- [Optional] Linear Objective Function
- Example:
  - maximize \( 4 x_4 + 3 x_{10} \)
  - minimize \( 3 x_1 + 3 x_3 \)
  - maximize \( 2 P + E \)
- Goal
  - Find values for \( x_1, \ldots, x_n \) satisfying all constraints, and
  - Maximize the objective
- Feasibility Problem
  - No objective function

Network Flow as a Linear Program

Given a directed graph \( G \) with capacities \( c(e) \) on each edge \( e \) we can use linear programming to find a maximum flow from source \( s \) to sink \( t \).

Variables: \( x_e \) for each directed edge \( e \) (represents flow on edge \( e \))

Objective: Maximize \( \sum_{e \text{ out of } s} x_e \)

Constraints:
- (Capacity Constraints) For each edge \( e \) we have \( 0 \leq x_e \leq c(e) \)
- (Flow Conservation) For each \( v \neq (s, t) \) we have
  \[ \sum_{e \text{ out of } v} x_e = \sum_{e \text{ into } v} x_e \]

Network Flow as a Linear Program

Example:

Variables: \( x_{a4}, x_{a2}, x_{a2}, x_{b2}, x_{a4} \)

Goal: maximize \( x_{a4} + x_{a2} \)

Subject to
- \( 0 \leq x_{a4} \leq 110 \)
- \( 0 \leq x_{a2} \leq 122 \)
- \( 0 \leq x_{23} \leq 1 \)
- \( 0 \leq x_{42} \leq 170 \)
- \( 0 \leq x_{44} \leq 102 \)
- \( x_{a4} = x_{a2} + x_{a4} \) [Flow Conservation at node 4]
- \( x_{a4} + x_{a2} = x_{a4} \) [Flow Conservation at node 2]

Linear Program Example

Goal: Maximize \( 2P+E \)

Subject to:
- [168 Hours] \( S + P + E = 168 \)
- [Maintain Sanity] \( P + E \geq 70 \)
- [Survive] \( E \geq 56 \)
- [Pass Courses 1] \( S \geq 60 \)
- [Pass Courses 2] \( 2S + E = 3P \geq 150 \)
- [Non-Negativity] \( P \geq 0 \)

Requirement:
- All the constrains are linear inequalities in variables \( (S, P, E) \)
- The objective function is also linear

Example Non-Linear Constraints:
\[
PE \geq 70 \quad E \in \{0, 1\} \\
P(1-E) = 1 \quad \text{Max}(P, E) \geq 20
\]

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

Credit for Example: Avrim Blum
Algorithmic Idea: Direction of Goodness

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

Worse Solution: $\vec{x}$
$c^T \cdot (\vec{x} - \vec{y}) < 0$

Initial Feasible Point: $\vec{x}_0$

Improved Solution: $\vec{y}$
$c^T \cdot (\vec{y} - \vec{x}_0) < 0$

Linear Programming

Theorem: Maximum value achieved at vertex (extreme point)

Definition: Let $F$ be the set of all feasible points in a linear program. We say that a point $p \in F$ is an extreme point (vertex) if every line segment $L \subset F$ that lies completely in $F$ 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 $2x_1 + 3x_2$  \( c=(2,3) \)

Initial Feasible Point: \( x_0 \)
Improved Solution: \( \bar{x} \)
\( \bar{x}^T \cdot (\bar{x} - x_0) < 0 \)

Feasible Point: \( x_1 \)

Optimal Point: \( x_2 \)
Worse Solution: \( \bar{y} \)
\( \bar{y}^T \cdot (\bar{y} - x_0) < 0 \)

Ellipsoid Algorithm: Solves Feasibility Problem

Step 1: Find large ellipse containing feasible region

Case 1: Center of ellipse is in \( F \)
Large Ellipse \( E \) Containing feasible region \( F \subset E \)

Case 2: Center of ellipse not in \( F \)
Every \( n \) steps volume drops by factor \( \frac{1}{e} \)
\( poly(n) \) iterations to find feasible point (or reject)

Can find smaller ellipsoid containing \( F \) smaller by at least a \( \left( 1 - \frac{1}{e} \right) \) factor.
Finding the Optimal Point with Ellipsoid Algorithm

**Goal:** maximize $\sum w_i x_i$ (where each $w_i$ is a constant)

**Key Idea:** Binary Search for value of Optimal Solution!

- Add Constraint $\sum w_i x_i \geq B$
- Infeasible? $\rightarrow$ Value of optimal solutions is less than $B$
- Feasible? $\rightarrow$ Value of optimal solution is at least $B$

Linear Programming in Practice

Many optimization packages available
- Solver (in Excel)
- LINDO
- CPLEX
- GUROBI (free academic license available)
- Matlab, Mathematica

More Linear Programming Examples

Brewer's Problem: Maximize Profit
- $(1$ Barrel) of Ale sells for $15$, but recipe requires
  - 6 pounds corn,
  - 5 ounces of hops and
  - 33 pounds of malt.
- $(1$ Barrel) of Beer sells for $27$, but recipe requires
  - 16 pounds of corn
  - 4 ounces of hops and
  - 21 pounds of malt
- Suppose we start off with $C=480$ pounds of corn, $H=160$ ounces of hops and $M=1190$ pounds of malt.
- Let $A$ (resp. $B$) denote number of barrels of Ale (resp. Beer)

**Goal:** maximize $15A+27B$

Solving in Mathematica

Maximize[$15 A + 27 B, A=0, B=0, 6 A + 16 B \leq 480, 5 A + 4 B \leq 160, 33 A + 21 B \leq 1190, \{A,B\}]

$(6060/7,\{A\to 80/7, B\to 180/7\})$

Profit: $865.71$
### 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>(0,0)</td>
<td>(-1,1)</td>
<td>(1,-1)</td>
</tr>
<tr>
<td>Paper</td>
<td>(1,-1)</td>
<td>(0,0)</td>
<td>(-1,1)</td>
</tr>
<tr>
<td>Scissors</td>
<td>(1,-1)</td>
<td>(1,-1)</td>
<td>(0,0)</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.

### Finding Minimax Optimal Solution using Linear Programming

**Variables:** \( p_1, \ldots, p_n \) and \( v \) (\( p_i \) is probability of action \( i \))

**Goal:** Maximize \( v \) (our expected reward).

**Constraints:**
- \( p_1, \ldots, p_n \geq 0 \)
- \( p_1 + \ldots + p_n \geq 0 \)
- For all columns \( j \) we have

\[
\sum_i p_{ij} \geq v
\]

\( m_{ij} \) denotes reward when player 1 takes action \( i \) and player 2 takes action \( j \).

### Extra Slides

**Circulation with Demands**

- **Directed graph** \( G = (V, E) \).
- **Edge capacities** \( c(e), e \in E \).
- **Node supply and demands** \( d(v), v \in V \).

**Def.** A circulation is a function that satisfies:

- For each \( e \in E \):
  \[
  0 \leq f(e) \leq c(e) \quad \text{(capacity)}
  \]
- For each \( v \in V \):
  \[
  \sum_{e \in E_v^+} f(e) - \sum_{e \in E_v^-} f(e) = d(v) \quad \text{(conservation)}
  \]

**Circulation problem:** given \( (V, E, c, d) \), does there exist a circulation?
Necessary condition: sum of supplies = sum of demands.
\[ \sum_{v : d(v) > 0} d(v) = \sum_{v : d(v) < 0} -d(v) = D \]

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

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

Pf. Follows from max flow formulation and integrity theorem for max flow.

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

\[ \sum_{v \in B} d(v) > \operatorname{cap}(A, B) \]

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

Circulation with Demands and Lower Bounds

Feasible circulation.
- Directed graph \( G = (V, E) \).
- Edge capacities \( c(e) \) and lower bounds \( l(e) \), \( e \in E \).
- Node supply and demands \( d(v) \), \( v \in V \).

Def. A circulation is a function that satisfies:
- For each \( e \in E \):
  \[ l(e) \leq f(e) \leq c(e) \] (capacity)
- For each \( v \in V \):
  \[ \sum_{e \in \text{in } v} f(e) - \sum_{e \in \text{out } v} f(e) = d(v) \] (conservation)

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

Idea. Model lower bounds with demands.
- Send \( l(e) \) 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.

Pf sketch. \( f(e) \) is a circulation in \( G \) iff \( f'(e) = f(e) - l(e) \) is a circulation in \( G' \).
### 7.8 Survey Design

**Survey design.**
- Design survey asking \( n_1 \) consumers about \( n_2 \) products.
- Can only survey consumer \( i \) about product \( j \) if they own it.
- Ask consumer \( i \) between \( c_i \) and \( c_i' \) questions.
- Ask between \( p_j \) and \( p_j' \) consumers about product \( j \).

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

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

### 7.10 Image Segmentation

**Image segmentation.**
- Central problem in image processing.
- Divide image into coherent regions.

**Ex:** Three people standing in front of complex background scene.
Identify each person as a coherent object.

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

### Foreground / background segmentation.
- Label each pixel in picture as belonging to foreground or background.
- \( V \) set of pixels, \( E \) pairs of neighboring pixels.
- \( a_i \geq 0 \) is likelihood pixel \( i \) in foreground.
- \( b_j \geq 0 \) is likelihood pixel \( j \) in 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:

\[
\sum_{i \in A} a_i + \sum_{j \in B} b_j - \sum_{\alpha \in A, \beta \in B} p_{\alpha \beta} \left[ \left| \alpha / \beta \right| \right]^{-1}
\]
Image Segmentation

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

Turn into minimization problem.
- Maximizing \( \sum_{i \in A} a_i + \sum_{j \in B} b_j - \sum_{(i,j) \in E} p_{ij} \)
- or alternatively \( \sum_{j \in B} b_j + \sum_{(i,j) \in E} p_{ij} \)

\( \text{or alternatively} \)

\[ \sum_{j \in B} b_j + \sum_{(i,j) \in E} p_{ij} \]

\[ \sum_{i \in A} a_i + \sum_{j \in B} b_j \]

Use two anti-parallel edges instead of undirected edge.

\[ G' = (V', E'). \]

Add source to correspond to foreground; add sink to correspond to background.

\[ s \rightarrow t \]

Consider min cut \((A, B)\) in \(G'\).
- \( A \) = foreground.

\[ \text{cap}(A, B) = \sum_{i \in A} a_i + \sum_{j \not\in A} b_j + \sum_{(i,j) \in E} p_{ij} \]

\( \text{if } i \text{ and } j \text{ on different sides, } \)

\( p_{ij} \text{ counted exactly once} \)

Precisely the quantity we want to minimize.

\[ G' \]
**Project Selection: Min Cut Formulation**

**Min cut formulation.**
- Assign capacity \( f \) to all prerequisite edge.
- Add edge \((s, v)\) with capacity \( p_v \) if \( p_v > 0 \).
- Add edge \((v, t)\) with capacity \(-p_v\) if \( p_v < 0 \).
- For notational convenience, define \( p_s = p_t = 0 \).

---

**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 \( p_v \) = value of ore - processing cost.
- Can’t remove block \( v \) before \( w \) or \( x \).

---

**Census Tabulation (Exercise 7.39)**

**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.

<table>
<thead>
<tr>
<th>Original matrix</th>
<th>Feasible rounding</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.14 6.8 7.3 17.24</td>
<td>3 7 7 17</td>
</tr>
<tr>
<td>9.6 2.4 0.7 12.7</td>
<td>10 2 1 13</td>
</tr>
<tr>
<td>3.6 1.2 6.5 11.3</td>
<td>3 1 7 11</td>
</tr>
<tr>
<td>16.34 10.4 14.5</td>
<td>16 10 15</td>
</tr>
</tbody>
</table>
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 $d_{ij}$, $a_i$, $b_j$ 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.

![Original matrix and feasible rounding]

**Theorem.** Feasible matrix rounding always exists.
**Pf.** Formulate as a circulation problem with lower bounds.
- Original data provides circulation (all demands = 0).
- Integrality theorem $\Rightarrow$ integral solution $\Rightarrow$ feasible rounding.