Linear Programming

- brewer’s problem
- simplex algorithm
- implementation
- linear programming

References:
The Allocation of Resources by Linear Programming, Scientific American, by Bob Bland
Algs in Java, Part 5
Overview: introduction to advanced topics

Main topics
- **linear programming**: the ultimate practical problem-solving model
- **reduction**: design algorithms, prove limits, classify problems
- **NP**: the ultimate theoretical problem-solving model
- **combinatorial search**: coping with intractability

Shifting gears
- from linear/quadratic to polynomial/exponential scale
- from individual problems to problem-solving models
- from details of implementation to conceptual framework

Goals
- place algorithms we’ve studied in a larger context
- introduce you to important and essential ideas
- inspire you to learn more about algorithms!
Linear Programming

What is it?

- Quintessential tool for optimal allocation of scarce resources, among a number of competing activities.
- Powerful and general problem-solving method that encompasses:
  - shortest path, network flow, MST, matching, assignment...
  - $Ax = b$, 2-person zero sum games

Why significant?

- Widely applicable problem-solving model
- Dominates world of industry.
- Fast commercial solvers available: CPLEX, OSL.
- Powerful modeling languages available: AMPL, GAMS.
- Ranked among most important scientific advances of 20th century.

Ex: Delta claims that LP saves $100 million per year.
Applications

_Agriculture._ Diet problem.
_Computer science._ Compiler register allocation, data mining.
_Electrical engineering._ VLSI design, optimal clocking.
_Energy._ Blending petroleum products.
_Economics._ Equilibrium theory, two-person zero-sum games.
_Environment._ Water quality management.
_Finance._ Portfolio optimization.
_Logistics._ Supply-chain management.
_Management._ Hotel yield management.
_Marketing._ Direct mail advertising.
_Manufacturing._ Production line balancing, cutting stock.
_Medicine._ Radioactive seed placement in cancer treatment.
_Operations research._ Airline crew assignment, vehicle routing.
_Physics._ Ground states of 3-D Ising spin glasses.
_Plasma physics._ Optimal stellarator design.
_Telecommunication._ Network design, Internet routing.
_Sports._ Scheduling ACC basketball, handicapping horse races.
brewer’s problem
simplex algorithm
implementation
linear programming
**Toy LP example: Brewer’s problem**

Small brewery produces ale and beer.

- Production limited by scarce resources: corn, hops, barley malt.
- Recipes for ale and beer require different proportions of resources.

<table>
<thead>
<tr>
<th></th>
<th>corn (lbs)</th>
<th>hops (oz)</th>
<th>malt (lbs)</th>
<th>profit ($)</th>
</tr>
</thead>
<tbody>
<tr>
<td>available</td>
<td>480</td>
<td>160</td>
<td>1190</td>
<td></td>
</tr>
<tr>
<td>ale (1 barrel)</td>
<td>5</td>
<td>4</td>
<td>35</td>
<td>13</td>
</tr>
<tr>
<td>beer (1 barrel)</td>
<td>15</td>
<td>4</td>
<td>20</td>
<td>23</td>
</tr>
</tbody>
</table>

**Brewer’s problem:** choose product mix to maximize profits.

<table>
<thead>
<tr>
<th></th>
<th>corn (lbs)</th>
<th>hops (oz)</th>
<th>malt (lbs)</th>
<th>profit ($)</th>
</tr>
</thead>
<tbody>
<tr>
<td>all ale (34 barrels)</td>
<td>179</td>
<td>136</td>
<td>1190</td>
<td>442</td>
</tr>
<tr>
<td>all beer (32 barrels)</td>
<td>480</td>
<td>128</td>
<td>640</td>
<td>736</td>
</tr>
<tr>
<td>20 barrels ale</td>
<td>400</td>
<td>160</td>
<td>1100</td>
<td>720</td>
</tr>
<tr>
<td>20 barrels beer</td>
<td>480</td>
<td>160</td>
<td>980</td>
<td>800</td>
</tr>
<tr>
<td>12 barrels ale</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>&gt;800?</td>
</tr>
<tr>
<td>28 barrels beer</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>more profitable product mix?</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>&gt;800?</td>
</tr>
</tbody>
</table>

34 barrels times 35 lbs malt per barrel is 1190 lbs [amount of available malt]
Brewer’s problem: mathematical formulation

Small brewery produces ale and beer.
• Production limited by scarce resources: corn, hops, barley malt.
• Recipes for ale and beer require different proportions of resources.

Mathematical formulation
• let \( A \) be the number of barrels of beer
• and \( B \) be the number of barrels of ale

\[
\text{maximize} \quad 13A + 23B \quad \text{profit}
\]

\[
\text{subject to the constraints} \quad 5A + 15B \leq 480 \quad \text{corn}
\]
\[
4A + 4B \leq 160 \quad \text{hops}
\]
\[
35A + 20B \leq 1190 \quad \text{malt}
\]
\[
A \geq 0 \\
B \geq 0
\]
Brewer’s problem: Feasible region

Hops
\[4A + 4B \leq 160\]

Malt
\[35A + 20B \leq 1190\]

Corn
\[5A + 15B \leq 480\]
Brewer’s problem: Objective function

\[ 13A + 23B = $800 \]
\[ 13A + 23B = $1600 \]
\[ 13A + 23B = $442 \]
Brewer’s problem: Geometry

Brewer’s problem observation. Regardless of objective function coefficients, an optimal solution occurs at an extreme point.
Standard form linear program

**Input:** real numbers $a_{ij}, c_j, b_i$.

**Output:** real numbers $x_j$.

$n = \# \text{nonnegative variables}, m = \# \text{constraints}.$

Maximize linear objective function subject to linear equations.

```
maximize $c_1 x_1 + c_2 x_2 + \ldots + c_n x_n$
subject to the
$m$ equations
$a_{11} x_1 + a_{12} x_2 + \ldots + a_{1n} x_n = b_1$
$a_{21} x_1 + a_{22} x_2 + \ldots + a_{2n} x_n = b_2$
...$
a_{m1} x_1 + a_{m2} x_2 + \ldots + a_{mn} x_n = b_m$
$x_1, x_2, \ldots, x_n \geq 0$
```

“Linear”

No $x^2, xy, \arccos(x)$, etc.

“Programming” “Planning” (term predates computer programming).
Converting the brewer’s problem to the standard form

Original formulation

maximize \( 13A + 23B \)
subject to the constraints
\[
\begin{align*}
5A &+ 15B &\leq 480 \\
4A &+ 4B &\leq 160 \\
35A &+ 20B &\leq 1190 \\
A, B & &\geq 0
\end{align*}
\]

Standard form

• add variable \( Z \) and equation corresponding to objective function
• add slack variable to convert each inequality to an equality.
• now a 5-dimensional problem.

maximize \( Z \)
subject to the constraints
\[
\begin{align*}
13A &+ 23B & \quad - Z &\quad = 0 \\
5A &+ 15B & + S_C &\quad = 480 \\
4A &+ 4B & + S_H &\quad = 160 \\
35A &+ 20B & + S_M &\quad = 1190 \\
A, B, S_C, S_H, S_M & & &\geq 0
\end{align*}
\]
Geometry

A few principles from geometry:
- inequality: halfplane (2D), hyperplane (kD).
- bounded feasible region: convex polygon (2D), convex polytope (kD).

Convex set. If two points $a$ and $b$ are in the set, then so is $\frac{1}{2}(a + b)$.

Extreme point. A point in the set that can't be written as $\frac{1}{2}(a + b)$, where $a$ and $b$ are two distinct points in the set.
Geometry (continued)

Extreme point property. If there exists an optimal solution to (P), then there exists one that is an extreme point.

Good news. Only need to consider finitely many possible solutions.

Bad news. Number of extreme points can be exponential!

Greedy property. Extreme point is optimal iff no neighboring extreme point is better.

Ex: n-dimensional hypercube

local optima are global optima
brewer’s problem
simplex algorithm
implementation
linear programming
Simplex Algorithm

**Simplex algorithm.** [George Dantzig, 1947]
- Developed shortly after WWII in response to logistical problems, including Berlin airlift.
- One of greatest and most successful algorithms of all time.

**Generic algorithm.**
- Start at some extreme point.
- **Pivot** from one extreme point to a neighboring one.
- Repeat until optimal.

**How to implement?** Linear algebra.
Simplex Algorithm: Basis

**Basis.** Subset of $m$ of the $n$ variables.

**Basic feasible solution (BFS).**
- Set $n - m$ nonbasic variables to 0, solve for remaining $m$ variables.
- Solve $m$ equations in $m$ unknowns.
- If unique and feasible solution $\Rightarrow$ BFS.
- BFS $\iff$ extreme point.

```
maximize      Z
subject to the constraints
13A + 23B    - Z = 0
5A + 15B + Sc = 480
4A + 4B + Sh = 160
35A + 20B + Sm = 1190
A, B, Sc, Sh, Sm \geq 0
```
Simplex Algorithm: Initialization

Start with slack variables as the basis.

Initial basic feasible solution (BFS).
- set non-basis variables $A = 0$, $B = 0$ (and $Z = 0$).
- 3 equations in 3 unknowns give $S_C = 480$, $S_C = 160$, $S_C = 1190$ (immediate).
- extreme point on simplex: origin

\[
\begin{align*}
\text{maximize} & \quad Z \\
\text{subject to the constraints} & \quad 13A + 23B - Z = 0 \\
& \quad 5A + 15B + S_C = 480 \\
& \quad 4A + 4B + S_H = 160 \\
& \quad 35A + 20B + S_M = 1190 \\
& \quad A, B, S_C, S_H, S_M \geq 0
\end{align*}
\]

basis = \{ $S_C$, $S_H$, $S_M$ \}
- $A = B = 0$
- $Z = 0$
- $S_C = 480$
- $S_H = 160$
- $S_M = 1190$
Simplex Algorithm: Pivot 1

maximize 
subject to the constraints

\[
\begin{align*}
Z & = 0 \\
13A + 23B & = SC \\
5A + 15B & = SH \\
4A + 4B & = SM \\
35A + 20B & = 0
\end{align*}
\]

which variable does it replace?

Substitution \( B = \frac{1}{15}(480 - 5A - SC) \) puts \( B \) into the basis
(rewrite 2nd equation, eliminate \( B \) in 1st, 3rd, and 4th equations)

maximize 
subject to the constraints

\[
\begin{align*}
Z & = -736 \\
\frac{16}{3}A - \frac{23}{15}SC & = 0 \\
\frac{1}{3}A + B + \frac{1}{15}SC & = 32 \\
\frac{8}{3}A - \frac{4}{15}SC + SH & = 32 \\
\frac{85}{3}A - \frac{4}{3}SC + SM & = 550
\end{align*}
\]

basis = \{SC, SH, SM\}
\begin{align*}
A &= B = 0 \\
Z &= 0 \\
SC &= 480 \\
SH &= 160 \\
SM &= 1190
\end{align*}
Simplex Algorithm: Pivot 1

maximize $Z$
subject to the constraints

<table>
<thead>
<tr>
<th></th>
<th>$13A$</th>
<th>$23B$</th>
<th>$-Z = 0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>$5A$</td>
<td>$15B$</td>
<td>$SC = 480$</td>
</tr>
<tr>
<td>$B$</td>
<td>$4A$</td>
<td>$4B$</td>
<td>$SH = 160$</td>
</tr>
<tr>
<td>$C$</td>
<td>$35A$</td>
<td>$20B$</td>
<td>$SM = 1190$</td>
</tr>
</tbody>
</table>

$A, B, SC, SH, SM \geq 0$

basis = \{SC, SH, SM\}

$A = B = 0$
$Z = 0$
$SC = 480$
$SH = 160$
$SM = 1190$

Why pivot on B?
• Its objective function coefficient is positive
  (each unit increase in B from 0 increases objective value by $23$)
• Pivoting on column 1 also OK.

Why pivot on row 2?
• Preserves feasibility by ensuring RHS $\geq 0$.
• Minimum ratio rule: $\min \{ 480/15, 160/4, 1190/20 \}.$
**Simplex Algorithm: Pivot 2**

maximize \[ Z \]
subject to the constraints

\[
\begin{align*}
(16/3)A - (23/15) SC & - Z = -736 \\
(1/3) A + B + (1/15) SC & = 32 \\
(8/3) A - (4/15) SC + SH & = 32 \\
(85/3) A - (4/3) SC + SM & = 550 \\
A, B, SC, SH, SM & \geq 0
\end{align*}
\]

basis = \{B, SH, SM\}

\[
\begin{align*}
A &= SC = 0 \\
Z &= 736 \\
B &= 32 \\
SH &= 32 \\
SM &= 550
\end{align*}
\]

Substitution \[ A = (3/8)(32 + (4/15) SC - SH) \] puts \[ A \] into the basis

(rewrite 3rd equation, eliminate \[ A \] in 1st, 2nd, and 4th equations)

maximize \[ Z \]
subject to the constraints

\[
\begin{align*}
& - SC - 2SH - Z = -800 \\
B + (1/10) SC + (1/8) SH & = 28 \\
A - (1/10) SC + (3/8) SH & = 12 \\
& - (25/6) SC - (85/8) SH + SM = 110 \\
A, B, SC, SH, SM & \geq 0
\end{align*}
\]

basis = \{A, B, SM\}

\[
\begin{align*}
SC &= SH = 0 \\
Z &= 800 \\
B &= 28 \\
A &= 12 \\
SM &= 110
\end{align*}
\]
Simplex algorithm: Optimality

Q. When to stop pivoting?
A. When all coefficients in top row are non-positive.

Q. Why is resulting solution optimal?
A. Any feasible solution satisfies system of equations in tableaux.
   • In particular: \( Z = 800 - S_C - 2S_H \)
   • Thus, optimal objective value \( Z^* \leq 800 \) since \( S_C, S_H \geq 0 \).
   • Current BFS has value 800 \( \Rightarrow \) optimal.

maximize \( Z \)
subject to the constraints

\[
\begin{align*}
B &\quad + \quad (1/10) \ S_C &\quad + \quad (1/8) \ S_H \quad = \quad 28 \\
A &\quad - \quad (1/10) \ S_C &\quad + \quad (3/8) \ S_H \quad = \quad 12 \\
&\quad - \quad (25/6) \ S_C &\quad - \quad (85/8) \ S_H &\quad + \quad S_M \quad = \quad 110 \\
A, B, S_C, S_H, S_M &\quad \geq \quad 0
\end{align*}
\]

basis = \{A, B, S_M\}
\[
\begin{align*}
S_C &\quad = \quad S_H \quad = \quad 0 \\
Z &\quad = \quad 800 \\
B &\quad = \quad 28 \\
A &\quad = \quad 12 \\
S_M &\quad = \quad 110
\end{align*}
\]
brewer’s problem
simplex algorithm
implementation
linear programming
Encode standard form LP in a single Java 2D array

Simplex tableau

maximize \[ Z \]
subject to the constraints

\[ 13A + 23B - Z = 0 \]
\[ 5A + 15B + S_C = 480 \]
\[ 4A + 4B + S_H = 160 \]
\[ 35A + 20B + S_M = 1190 \]
\[ A, B, S_C, S_H, S_M \geq 0 \]

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>I</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>m</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>c</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>n</td>
<td></td>
<td>m</td>
<td>1</td>
</tr>
</tbody>
</table>

\[
\begin{bmatrix}
5 & 15 & 1 & 0 & 0 & 480 \\
4 & 4 & 0 & 1 & 0 & 160 \\
35 & 20 & 0 & 0 & 1 & 1190 \\
13 & 23 & 0 & 0 & 0 & 0
\end{bmatrix}
\]
**Simplex tableau**

**Encode standard form LP in a single Java 2D array (solution)**

<table>
<thead>
<tr>
<th>maximize</th>
<th>( Z )</th>
</tr>
</thead>
<tbody>
<tr>
<td>subject</td>
<td>( - \quad S_c \quad - \quad 2S_H \quad - \quad Z = -800 )</td>
</tr>
<tr>
<td>to the</td>
<td>( B \quad + \quad (1/10) \quad S_c \quad + \quad (1/8) \quad S_H \quad = \quad 28 )</td>
</tr>
<tr>
<td>constraints</td>
<td>( A \quad - \quad (1/10) \quad S_c \quad + \quad (3/8) \quad S_H \quad = \quad 12 )</td>
</tr>
<tr>
<td></td>
<td>( - \quad (25/6) \quad S_c \quad - \quad (85/8) \quad S_H \quad + \quad S_M \quad = \quad 110 )</td>
</tr>
<tr>
<td></td>
<td>( A, B, S_c, S_H, S_M \quad \geq \quad 0 )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>1/10</th>
<th>1/8</th>
<th>0</th>
<th>28</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1/10</td>
<td>3/8</td>
<td>0</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>25/6</td>
<td>85/8</td>
<td>1</td>
<td>110</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>-2</td>
<td>0</td>
<td>-800</td>
<td></td>
</tr>
</tbody>
</table>

**Simplex algorithm** transforms initial array into solution
**Simplex algorithm: Bare-bones implementation**

**Construct the simplex tableau.**

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>I</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>c</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

```
public class Simplex
{
    private double[][] a; // simplex tableaux
    private int M, N;

    public Simplex(double[][] A, double[] b, double[] c)
    {
        M = b.length;
        N = c.length;
        a = new double[M+1][M+N+1];
        for (int i = 0; i < M; i++)
            for (int j = 0; j < N; j++)
                a[i][j] = A[i][j];
        for (int j = N; j < M + N; j++)
            a[j-N][j] = 1.0;
        for (int j = 0; j < N; j++)
            a[M][j] = c[j];
        for (int i = 0; i < M; i++)
            a[i][M+N] = b[i];
    }
}
```
Simplex algorithm: Bare-bones Implementation

Pivot on element \((p, q)\).

```java
public void pivot(int p, int q)
{
    for (int i = 0; i <= M; i++)
        for (int j = 0; j <= M + N; j++)
            if (i != p && j != q)
                a[i][j] -= a[p][j] * a[i][q] / a[p][q];
    for (int i = 0; i <= M; i++)
        if (i != p) a[i][q] = 0.0;
    for (int j = 0; j <= M + N; j++)
        if (j != q) a[p][j] /= a[p][q];
    a[p][q] = 1.0;
}
```

- scale all elements but row \(p\) and column \(q\)
- zero out column \(q\)
- scale row \(p\)
Simplex Algorithm: Bare Bones Implementation

Simplex algorithm.

```java
public void solve()
{
    while (true)
    {
        int p, q;
        for (q = 0; q < M + N; q++)
            if (a[M][q] > 0) break;
        if (q >= M + N) break;
        for (p = 0; p < M; p++)
            if (a[p][q] > 0) break;
        for (int i = p+1; i < M; i++)
            if (a[i][q] > 0)
                if (a[i][M+N] / a[i][q] < a[p][M+N] / a[p][q])
                    p = i;
        pivot(p, q);
    }
}
```

- **find entering variable q** (positive objective function coefficient)
- **find row p according to min ratio rule**
- **min ratio test**
Remarkable property. In practice, simplex algorithm typically terminates after at most $2(m+n)$ pivots.

- No pivot rule that is guaranteed to be polynomial is known.
- Most pivot rules known to be exponential (or worse) in worst-case.

Pivoting rules. Carefully balance the cost of finding an entering variable with the number of pivots needed.
Simplex algorithm: Degeneracy

**Degeneracy.** New basis, same extreme point.

"stalling" is common in practice

**Cycling.** Get stuck by cycling through different bases that all correspond to same extreme point.
- Doesn't occur in the wild.
- Bland's least index rule guarantees finite # of pivots.
Simplex Algorithm: Implementation Issues

To improve the bare-bones implementation
- Avoid stalling.
- Choose the pivot wisely.
- Watch for numerical stability.
- Maintain sparsity.  \[\text{requires fancy data structures}\]
- Detect infeasiblity
- Detect unboundedness.
- Preprocess to reduce problem size.

Basic implementations available in many programming environments.

Commercial solvers routinely solve LPs with millions of variables.
import drasys.or.mp.*;
import drasys.or.mp.lp.*;

public class LPDemo
{
    public static void main(String[] args) throws Exception
    {
        Problem prob = new Problem(3, 2);
        prob.getMetadata().put("lp.isMaximize", "true");
        prob.newVariable("x1").setObjectiveCoefficient(13.0);
        prob.newVariable("x2").setObjectiveCoefficient(23.0);
        prob.newConstraint("corn").setRightHandSide(480.0);
        prob.newConstraint("hops").setRightHandSide(160.0);
        prob.newConstraint("malt").setRightHandSide(1190.0);

        prob.setCoefficientAt("corn", "x1", 5.0);
        prob.setCoefficientAt("corn", "x2", 15.0);
        prob.setCoefficientAt("hops", "x1", 4.0);
        prob.setCoefficientAt("hops", "x2", 4.0);
        prob.setCoefficientAt("malt", "x1", 35.0);
        prob.setCoefficientAt("malt", "x2", 20.0);

        DenseSimplex lp = new DenseSimplex(prob);
        System.out.println(lp.solve());
        System.out.println(lp.getSolution());
    }
}
**LP solvers: commercial strength**

**AMPL.** [Fourer, Gay, Kernighan] An algebraic modeling language.

**CPLEX solver.** Industrial strength solver.

---

**Beer Production Problem**

- **Ingredients (INGR):**
  - Corn
  - Hops
  - Malt

- **Products (PROD):**
  - Ale
  - Beer

- **Profit Parameters:**
  - Ale: 13
  - Beer: 23

- **Supply Parameters:**
  - Corn: 480
  - Hops: 160
  - Malt: 1190

**Objective:**

Maximize total profit:

\[
\text{profit} = 13A + 23B
\]

Subject to constraints:

\[
\begin{align*}
5A + 15B & \leq 480 \\
4A + 4B & \leq 160 \\
35A + 20B & \leq 1190
\end{align*}
\]

- \(A \geq 0\)
- \(B \geq 0\)

---

**AMPL Model (beer.mod):**

```plaintext
set INGR;
set PROD;
param profit {PROD};
param supply {INGR};
param amt {INGR, PROD};
var x {PROD} >= 0;
maximize total_profit:
    sum {j in PROD} x[j] * profit[j];
subject to constraints {i in INGR}:
    sum {j in PROD} amt[i,j] * x[j] <= supply[i];
```

---

**Separate Data from Model**

**Data File (beer.dat):**

```plaintext
set PROD := beer ale;
set INGR := corn hops malt;
param: profit :=
    ale 13
    beer 23;
param: supply :=
    corn 480
    hops 160
    malt 1190;
p
```
History

1939. Production, planning. [Kantorovich]
1947. Simplex algorithm. [Dantzig]
1950. Applications in many fields.
1979. Ellipsoid algorithm. [Khachian]
1984. Projective scaling algorithm. [Karmarkar]
1990. Interior point methods.
   • Interior point faster when polyhedron smooth like disco ball.
   • Simplex faster when polyhedron spiky like quartz crystal.

200x. Approximation algorithms, large scale optimization.
Linear programming

Linear “programming”
• process of formulating an LP model for a problem
• solution to LP for a specific problem gives solution to the problem

1. Identify variables
2. Define constraints (inequalities and equations)
3. Define objective function

Examples:
• shortest paths
• maxflow
• bipartite matching
• .
• .
• .
• [ a very long list ]

easy part [omitted]: convert to standard form

stay tuned [this lecture]
Single-source shortest-paths problem (revisited)

**Given.** Weighted digraph, single source $s$.

**Distance** from $s$ to $v$: length of the shortest path from $s$ to $v$.

**Goal.** Find distance (and shortest path) from $s$ to every other vertex.
LP formulation of single-source shortest-paths problem

One variable per vertex, one inequality per edge.

minimize $x_t$
subject to the constraints

interpretation: $x_i =$ length of shortest path from source to $i$

$x_s + 9 \leq x_2$
$x_s + 14 \leq x_6$
$x_s + 15 \leq x_7$
$x_2 + 24 \leq x_3$
$x_3 + 2 \leq x_5$
$x_3 + 19 \leq x_t$
$x_4 + 6 \leq x_3$
$x_4 + 6 \leq x_t$
$x_5 + 11 \leq x_4$
$x_5 + 16 \leq x_t$
$x_6 + 18 \leq x_3$
$x_6 + 30 \leq x_5$
$x_6 + 5 \leq x_7$
$x_7 + 20 \leq x_5$
$x_7 + 44 \leq x_t$
$x_s = 0$
$x_2 , \ldots , x_t \geq 0$
LP formulation of single-source shortest-paths problem

One variable per vertex, one inequality per edge.

minimize $x_t$
subject to the constraints

interpretation: $x_i$ = length of shortest path from source to $i$

solution

$x_s = 0$
$x_2 = 9$
$x_3 = 32$
$x_4 = 45$
$x_5 = 34$
$x_6 = 14$
$x_7 = 15$
$x_t = 50$
Maxflow problem

**Given:** Weighted digraph, source \( s \), destination \( t \).

Interpret edge weights as **capacities**
- Models material flowing through network
- Ex: oil flowing through pipes
- Ex: goods in trucks on roads
- [many other examples]

**Flow:** A different set of edge weights
- flow does not exceed capacity in any edge
- flow at every vertex satisfies **equilibrium**
  [flow in equals flow out]

**Goal:** Find **maximum flow** from \( s \) to \( t \)
LP formulation of maxflow problem

One variable per edge.
One inequality per edge, one equality per vertex.

maximize \( x_{ts} \)
subject to the constraints
\[
\begin{align*}
x_{s1} & \leq 2 \\
x_{s2} & \leq 3 \\
x_{13} & \leq 3 \\
x_{14} & \leq 1 \\
x_{23} & \leq 1 \\
x_{24} & \leq 1 \\
x_{3t} & \leq 2 \\
x_{4t} & \leq 3
\end{align*}
\]

interpretation: \( x_{ij} = \text{flow in edge } i-j \)
equilibrium constraints
\[
\begin{align*}
x_{ts} &= x_{s1} + x_{s2} \\
x_{s1} &= x_{13} + x_{14} \\
x_{s2} &= x_{23} + x_{24} \\
x_{13} + x_{23} &= x_{3t} \\
x_{14} + x_{24} &= x_{4t} \\
x_{3t} + x_{4t} &= x_{ts} \\
\text{all } x_{ij} &\geq 0
\end{align*}
\]
capacity constraints

add dummy edge from \( t \) to \( s \)
LP formulation of maxflow problem

One variable per edge.
One inequality per edge, one equality per vertex.

**maximize** \( x_{ts} \)

subject to the constraints

- \( x_{s1} \leq 2 \)
- \( x_{s2} \leq 3 \)
- \( x_{13} \leq 3 \)
- \( x_{14} \leq 1 \)
- \( x_{23} \leq 1 \)
- \( x_{24} \leq 1 \)
- \( x_{3t} \leq 2 \)
- \( x_{4t} \leq 3 \)

**equilibrium constraints**

\[
\begin{align*}
X_{ts} &= X_{s1} + X_{s2} \\
X_{s1} &= X_{13} + X_{14} \\
X_{s2} &= X_{23} + X_{24} \\
X_{13} + X_{23} &= X_{3t} \\
X_{14} + X_{24} &= X_{4t} \\
X_{3t} + X_{4t} &= X_{ts} \\
\end{align*}
\]

**interpretation:** \( x_{ij} = \text{flow in edge } i-j \)

add dummy edge from \( t \) to \( s \)

**solution**

- \( X_{s1} = 2 \)
- \( X_{s2} = 2 \)
- \( X_{13} = 1 \)
- \( X_{14} = 1 \)
- \( X_{23} = 1 \)
- \( X_{24} = 1 \)
- \( X_{3t} = 2 \)
- \( X_{4t} = 2 \)
- \( X_{ts} = 4 \)

**maxflow value**
Maximum cardinality bipartite matching problem

**Given:** Two sets of vertices, set of edges
(each connecting one vertex in each set)

**Matching:** set of edges
with no vertex appearing twice

**Interpretation:** mutual preference constraints
- Ex: people to jobs
- Ex: medical students to residence positions
- Ex: students to writing seminars
- [many other examples]

**Goal:** find a maximum cardinality matching

Example: Job offers
LP formulation of maximum cardinality bipartite matching problem

One variable per edge, one equality per vertex.

\[\begin{align*}
\text{maximize} & \quad x_{A0} + x_{A1} + x_{A2} + x_{B0} + x_{B1} + x_{B5} \\
& \quad + x_{C2} + x_{C3} + x_{C4} + x_{D0} + x_{D1} \\
& \quad + x_{E3} + x_{E4} + x_{E5} + x_{F2} + x_{F4} + x_{F5} \\
\text{subject to the constraints} & \quad x_{A0} + x_{A1} + x_{A2} = 1 \\
& \quad x_{B0} + x_{B1} + x_{B5} = 1 \\
& \quad x_{C2} + x_{C3} + x_{C4} = 1 \\
& \quad x_{D0} + x_{D1} = 1 \\
& \quad x_{E3} + x_{E4} + x_{E5} = 1 \\
& \quad x_{F2} + x_{F4} + x_{F5} = 1 \\
& \quad x_{A0} + x_{B0} + x_{D0} = 1 \\
& \quad x_{A1} + x_{B1} + x_{D1} = 1 \\
& \quad x_{A2} + x_{C2} + x_{F2} = 1 \\
& \quad x_{C3} + x_{E3} = 1 \\
& \quad x_{C4} + x_{E4} + x_{F4} = 1 \\
& \quad x_{B5} + x_{E5} + x_{F5} = 1 \\
& \quad \text{all } x_{ij} \geq 0
\end{align*}\]

**Theorem.** [Birkhoff 1946, von Neumann 1953] All extreme points of the above polyhedron have integer (0 or 1) coordinates.

**Corollary.** Can solve bipartite matching problem by solving LP.
LP formulation of maximum cardinality bipartite matching problem

One variable per edge, one equality per vertex.

maximize \( x_{A0} + x_{A1} + x_{A2} + x_{B0} + x_{B1} + x_{B5} + x_{C2} + x_{C3} + x_{C4} + x_{D0} + x_{D1} + x_{E3} + x_{E4} + x_{E5} + x_{F2} + x_{F4} + x_{F5} \)

subject to the constraints

\begin{align*}
  x_{A0} + x_{A1} + x_{A2} &= 1 \\
  x_{B0} + x_{B1} + x_{B5} &= 1 \\
  x_{C2} + x_{C3} + x_{C4} &= 1 \\
  x_{D0} + x_{D1} &= 1 \\
  x_{E3} + x_{E4} + x_{E5} &= 1 \\
  x_{F2} + x_{F4} + x_{F5} &= 1 \\
  x_{A0} + x_{B0} + x_{D0} &= 1 \\
  x_{A1} + x_{B1} + x_{D1} &= 1 \\
  x_{A2} + x_{C2} + x_{F2} &= 1 \\
  x_{C3} + x_{E3} &= 1 \\
  x_{C4} + x_{E4} + x_{F4} &= 1 \\
  x_{B5} + x_{E5} + x_{F5} &= 1 \\
  \text{all } x_{ij} &\geq 0
\end{align*}

interpretation: An edge is in the matching iff \( x_{ij} = 1 \)

solution

\begin{align*}
  x_{A1} &= 1 \\
  x_{B5} &= 1 \\
  x_{C2} &= 1 \\
  x_{D0} &= 1 \\
  x_{E3} &= 1 \\
  x_{F4} &= 1 \\
  \text{all other } x_{ij} &= 0
\end{align*}
Got an optimization problem?
  ex: shortest paths, maxflow, matching, ... [many, many, more]

Approach 1: Use a specialized algorithm to solve it
  • Algs in Java
  • vast literature on complexity
  • performance on real problems not always well-understood

Approach 2: Use linear programming
  • a direct mathematical representation of the problem often works
  • immediate solution to the problem at hand is often available
  • might miss specialized solution, but might not care

Got an LP solver? Learn to use it!

[cos226:tucson] ~> ampl
AMPL Version 20010215 (SunOS 5.7)
ampl: model maxflow.mod;
ampl: data maxflow.dat;
ampl: solve;
CPLEX 7.1.0: optimal solution;
objective 4;
LP: the ultimate problem-solving model (in practice)

Fact 1: Many practical problems are easily formulated as LPs
Fact 2: Commercial solvers can solve those LPs quickly

More constraints on the problem?
- specialized algorithm may be hard to fix
- can just add more inequalities to LP

New problem?
- may not be difficult to formulate LP
- may be very difficult to develop specialized algorithm

Today’s problem?
- similar to yesterday’s
- edit tableau, run solver

Too slow?
- could happen
- doesn’t happen

Ex. Mincost maxflow and other generalized versions
Ex. Airline scheduling
[ similar to vast number of other business processes ]

Want to learn more?
ORFE 307
Is there an ultimate problem-solving model?

- Shortest paths
- Maximum flow
- Bipartite matching
- ...
- Linear programming

\{
\text{tractable}
\}

- \ldots
- NP-complete problems

\{
\text{intractable ?}
\}

\text{Does P = NP?} \quad \text{No universal problem-solving model exists unless P = NP.}

[see next lecture]
**LP perspective**

LP is near the deep waters of intractability.

**Good news:**
- LP has been widely used for large practical problems for 50+ years
- Existence of guaranteed poly-time algorithm known for 25+ years.

**Bad news:**
- Integer linear programming is NP-complete
- (existence of guaranteed poly-time algorithm is highly unlikely).
- [stay tuned]

An unsuspecting MBA student transitions to the world of intractability with a single mouse click.