6. Linear Programming

- brewer’s problem
- simplex algorithm
- implementations
- duality
- modeling
Overview: introduction to advanced topics

Main topics. [next 3 lectures]

• Linear programming: the ultimate practical problem-solving model.
• NP: the ultimate theoretical problem-solving model.
• Reduction: design algorithms, establish lower bounds, classify problems.
• Combinatorial search: coping with intractability.

Shifting gears.

• From individual problems to problem-solving models.
• From linear/quadratic to polynomial/exponential scale.
• 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 problem-solving model for optimal allocation of scarce resources, among a number of competing activities that encompasses:

- Shortest paths, maxflow, MST, matching, assignment, ...
- $Ax = b$, 2-person zero-sum games, ...

Why significant?

- Fast commercial solvers available.
- Widely applicable problem-solving model.
- Key subroutine for integer programming solvers.

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

maximize $13A + 23B$

subject to

- $5A + 15B \leq 480$
- $4A + 4B \leq 160$
- $35A + 20B \leq 1190$

$A, B \geq 0$

to learn much much more, see ORF 307
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.


Operations research. Airline crew assignment, vehicle routing.

Physics. Ground states of 3-D Ising spin glasses.

Telecommunication. Network design, Internet routing.

Sports. Scheduling ACC basketball, handicapping horse races.
The Allocation of Resources by Linear Programming by Robert Bland,
Scientific American, Vol. 244, No. 6, June 1981.
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>Resource</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>corn</td>
<td>480 lbs</td>
</tr>
<tr>
<td>hops</td>
<td>160 oz</td>
</tr>
<tr>
<td>malt</td>
<td>1190 lbs</td>
</tr>
</tbody>
</table>

- 5 pounds corn
- 4 ounces hops
- 35 pounds malt

$13 profit per barrel

- 5.5 pounds corn
- 4 ounces hops
- 20 pounds malt

$23 profit per barrel
**Toy LP example: brewer’s problem**

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

<table>
<thead>
<tr>
<th>ale</th>
<th>beer</th>
<th>corn</th>
<th>hops</th>
<th>malt</th>
<th>profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>34</td>
<td>0</td>
<td>179</td>
<td>136</td>
<td>1190</td>
<td>$442</td>
</tr>
<tr>
<td>0</td>
<td>32</td>
<td>480</td>
<td>128</td>
<td>640</td>
<td>$736</td>
</tr>
<tr>
<td>19.5</td>
<td>20.5</td>
<td>405</td>
<td>160</td>
<td>1092.5</td>
<td>$725</td>
</tr>
<tr>
<td>12</td>
<td>28</td>
<td>480</td>
<td>160</td>
<td>980</td>
<td>$800</td>
</tr>
<tr>
<td>?</td>
<td>?</td>
<td></td>
<td></td>
<td>&gt; $800</td>
<td>?</td>
</tr>
</tbody>
</table>

- 34 barrels × 35 lbs malt = 1190 lbs

- Amount of available malt

- Good are indivisible

- Corn: 480 lbs
- Hops: 160 oz
- Malt: 1190 lbs

- $13 profit per barrel
- $23 profit per barrel
Brewer’s problem: linear programming formulation

Linear programming formulation.

- Let $A$ be the number of barrels of ale.
- Let $B$ be the number of barrels of beer.

<table>
<thead>
<tr>
<th></th>
<th>ale</th>
<th>beer</th>
</tr>
</thead>
<tbody>
<tr>
<td>maximize</td>
<td>$13A$</td>
<td>$23B$</td>
</tr>
<tr>
<td>subject</td>
<td>$5A$</td>
<td>$15B$</td>
</tr>
<tr>
<td></td>
<td>$4A$</td>
<td>$4B$</td>
</tr>
<tr>
<td>constraints</td>
<td>$35A$</td>
<td>$20B$</td>
</tr>
<tr>
<td></td>
<td>$A$,</td>
<td>$B$</td>
</tr>
</tbody>
</table>

subject to the constraints:

- $5A + 15B \leq 480$
- $4A + 4B \leq 160$
- $35A + 20B \leq 1190$
- $A, B \geq 0$
Inequalities define **halfplanes**; feasible region is a **convex polygon**.

**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

profit

(0, 0)   (12, 28)   (26, 14)   (34, 0)

(0, 32)

beer

(13A + 23B = $1600)
(13A + 23B = $800)
(13A + 23B = $442)
Regardless of objective function, optimal solution occurs at an extreme point.

Brewer's problem: geometry

intersection of 2 constraints in 2d
**Goal.** Maximize linear objective function of \( n \) nonnegative variables, subject to \( m \) linear equations.

- **Input:** real numbers \( a_{ij}, c_j, b_i \).
- **Output:** real numbers \( x_j \).

### primal problem (P)

maximize \[ c_1 x_1 + c_2 x_2 + \ldots + c_n x_n \]
subject to the constraints \[ a_{11} x_1 + a_{12} x_2 + \ldots + a_{1n} x_n = b_1 \]
\[ a_{21} x_1 + a_{22} x_2 + \ldots + a_{2n} x_n = b_2 \]
\[ \vdots \]
\[ a_{m1} x_1 + a_{m2} x_2 + \ldots + a_{mn} x_n = b_m \]
\[ x_1, x_2, \ldots, x_n \geq 0 \]

### matrix version

maximize \[ c^T x \]
subject to the constraints \[ A x = b \]
\[ x \geq 0 \]

linear means no \( x^2 \), \( xy \), \( \arccos(x) \), etc.
Converting the brewer’s problem to the standard form

Original formulation.

<table>
<thead>
<tr>
<th>maximize</th>
<th>13A + 23B</th>
</tr>
</thead>
<tbody>
<tr>
<td>subject to the constraints</td>
<td></td>
</tr>
<tr>
<td>5A + 15B ≤ 480</td>
<td></td>
</tr>
<tr>
<td>4A + 4B ≤ 160</td>
<td></td>
</tr>
<tr>
<td>35A + 20B ≤ 1190</td>
<td></td>
</tr>
<tr>
<td>A, B ≥ 0</td>
<td></td>
</tr>
</tbody>
</table>

Standard form.

- Add variable $Z$ and equation corresponding to objective function.
- Add slack variable to convert each inequality to an equality.
- Now a 6-dimensional problem.

<table>
<thead>
<tr>
<th>maximize</th>
<th>$Z$</th>
</tr>
</thead>
<tbody>
<tr>
<td>subject to the constraints</td>
<td></td>
</tr>
<tr>
<td>13A + 23B</td>
<td></td>
</tr>
<tr>
<td>5A + 15B + $S_C$ = 480</td>
<td></td>
</tr>
<tr>
<td>4A + 4B + $S_H$ = 160</td>
<td></td>
</tr>
<tr>
<td>35A + 20B + $S_M$ = 1190</td>
<td></td>
</tr>
<tr>
<td>A, B, $S_C$, $S_C$, $S_M$ ≥ 0</td>
<td></td>
</tr>
</tbody>
</table>
Inequalities define halfspaces; feasible region is a convex polyhedron.

A set is convex if for any two points \(a\) and \(b\) in the set, so is \(\frac{1}{2} (a + b)\).

An extreme point of a set is 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.

Warning. Don't always trust intuition in higher dimensions.
Geometry (continued)

**Extreme point property.** If there exists an optimal solution to \((P)\), then there exists one that is an extreme point.
- Number of extreme points to consider is **finite**.
- But number of extreme points can be **exponential!**

**Greedy property.** Extreme point optimal iff no better adjacent extreme point.

Local optima are global optima (follows because objective function is linear and feasible region is convex)
› brewer’s problem
› simplex algorithm
› implementations
› duality
› modeling
Simplex algorithm

Simplex algorithm. [George Dantzig, 1947]
• Developed shortly after WWII in response to logistical problems, including Berlin airlift.
• Ranked as one of top 10 scientific algorithms of 20th century.

Generic algorithm.
• Start at some extreme point.
• Pivot from one extreme point to an adjacent one.
• Repeat until optimal.

How to implement? Linear algebra.
Simplex algorithm: basis

A basis is a 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 \( \Rightarrow \) BFS.
- BFS \( \iff \) extreme point.

<table>
<thead>
<tr>
<th>maximize</th>
<th>( Z )</th>
</tr>
</thead>
<tbody>
<tr>
<td>subject</td>
<td>( 13A + 23B - Z = 0 )</td>
</tr>
<tr>
<td>to the constraints</td>
<td>( 5A + 15B + S_C = 480 )</td>
</tr>
<tr>
<td>constraints</td>
<td>( 4A + 4B + S_H = 160 )</td>
</tr>
<tr>
<td></td>
<td>( 35A + 20B + S_M = 1190 )</td>
</tr>
<tr>
<td></td>
<td>( A, B, S_C, S_H, S_M \geq 0 )</td>
</tr>
</tbody>
</table>

\({A, B, S_C}\) (26, 14)
\({A, B, S_H}\) (19.41, 25.53)
\({A, S_H, S_C}\) (34, 0)
\({A, B, S_M}\) (12, 28)
\({S_H, S_M, S_C}\) (0, 0)
\({B, S_H, S_M}\) (0, 32)
### Simplex algorithm: initialization

**Initial basic feasible solution.**

- Start with slack variables \( \{ S_C, S_H, S_M \} \) as the basis.
- Set non-basic variables \( A \) and \( B \) to 0.
- 3 equations in 3 unknowns yields \( S_C = 480, S_H = 160, S_M = 1190 \).
### Simplex algorithm: pivot 1

<table>
<thead>
<tr>
<th>maximize</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>subject</td>
<td>13A + 23B</td>
</tr>
<tr>
<td>to the constraints</td>
<td>(-Z = 0)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>5A</th>
<th>15B</th>
<th>SC</th>
</tr>
</thead>
<tbody>
<tr>
<td>4A</td>
<td>4B</td>
<td>SH</td>
</tr>
<tr>
<td>35A</td>
<td>20B</td>
<td>SM</td>
</tr>
<tr>
<td>A</td>
<td>B</td>
<td>SC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SH</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SM</td>
</tr>
</tbody>
</table>

\[A = B = 0\]
\[Z = 0\]
\[SC = 480\]
\[SH = 160\]
\[SM = 1190\]

Substitute \(B = \frac{1}{15}(480 - 5A - SC)\) and add \(B\) into the basis.

Which basic variable does \(B\) replace?

<table>
<thead>
<tr>
<th>maximize</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>subject</td>
<td>((16/3)A) - ((23/15)SC)</td>
</tr>
<tr>
<td>to the constraints</td>
<td>(-Z = -736)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>((1/3)A)</th>
<th>B</th>
<th>((1/15)SC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>((8/3)A)</td>
<td>-</td>
<td>((4/15)SC) + SH</td>
</tr>
<tr>
<td>((85/3)A)</td>
<td>-</td>
<td>((4/3)SC) + SM</td>
</tr>
<tr>
<td>A, B, SC, SH, SM</td>
<td>≥ 0</td>
<td></td>
</tr>
</tbody>
</table>

\[basis = \{SC, SH, SM\}\]
\[A = S_C = 0\]
\[Z = 736\]
\[B = 32\]
\[S_H = 32\]
\[S_M = 550\]
**Simplex algorithm: pivot 1**

<table>
<thead>
<tr>
<th>maximize</th>
<th>Z</th>
<th>+</th>
<th>23B</th>
<th>pivot</th>
<th>subject to the constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>13A</td>
<td>-Z</td>
<td>=</td>
<td>0</td>
<td></td>
<td>15B + S_C</td>
</tr>
<tr>
<td>5A</td>
<td>5A</td>
<td>+</td>
<td>15B</td>
<td></td>
<td>480</td>
</tr>
<tr>
<td>4A</td>
<td>4A</td>
<td>+</td>
<td>4B</td>
<td></td>
<td>160</td>
</tr>
<tr>
<td>35A</td>
<td>35A</td>
<td>+</td>
<td>20B</td>
<td></td>
<td>1190</td>
</tr>
<tr>
<td>A, B, S_C, S_H, S_M</td>
<td>A, B, S_C, S_H, S_M</td>
<td>≥</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Q. Why pivot on column 2 (corresponding to variable B)?
- Its objective function coefficient is positive.
  (each unit increase in B from 0 increases objective value by $23)
- Pivoting on column 1 (corresponding to A) also OK.

Q. Why pivot on row 2?
- Preserves feasibility by ensuring RHS ≥ 0.
- Minimum ratio rule: min { 480/15, 160/4, 1190/20 }.

basis = \{ S_C, S_H, S_M \}
- A = B = 0
- Z = 0
- S_C = 480
- S_H = 160
- S_M = 1190
**Simplex algorithm: pivot 2**

maximize \[ Z \]
subject to the constraints

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

basis = \{ B, S_H, S_M \}

A = S_C = 0
Z = 736
B = 32
S_H = 32
S_M = 550

substitute \( A = (3/8) (32 + (4/15) S_C - S_H) \) and add \( A \) into the basis
(rewrite 3rd equation, eliminate \( A \) in 1st, 2nd, and 4th equations)

maximize \[ Z \]
subject to the constraints

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

basis = \{ A, B, S_M \}

S_C = S_H = 0
Z = 800
B = 28
A = 12
S_M = 110

which basic variable does \( A \) replace?
Simplex algorithm: optimality

**Q.** When to stop pivoting?

**A.** When no objective function coefficient is positive.

**Q.** Why is resulting solution optimal?

**A.** Any feasible solution satisfies current system of equations.

- In particular: \[ Z = 800 - S_C - 2 S_H \]
- Thus, optimal objective value \( Z^* \leq 800 \) since \( S_C, S_H \geq 0 \).
- Current BFS has value 800 \( \Rightarrow \) optimal.

<table>
<thead>
<tr>
<th>maximize</th>
<th>( Z )</th>
</tr>
</thead>
<tbody>
<tr>
<td>subject to the constraints</td>
<td>(-)</td>
</tr>
<tr>
<td>( B )</td>
<td>+</td>
</tr>
<tr>
<td>( A )</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>-</td>
</tr>
<tr>
<td>( A, B, S_C, S_H, S_M )</td>
<td>\geq</td>
</tr>
</tbody>
</table>

basis = \{ A, B, S_M \}

\( S_C = S_H = 0 \)

\( Z = 800 \)

\( B = 28 \)

\( A = 12 \)

\( S_M = 110 \)
Encode standard form LP in a single Java 2D array.

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>15</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>480</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>160</td>
</tr>
<tr>
<td>35</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1190</td>
</tr>
</tbody>
</table>

A, B, S_C, S_H, S_M ≥ 0

Initial simplex tableaux
Simplex tableau

Simplex algorithm transforms initial 2D array into solution.

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

\frac{1}{10} \ S_C + \frac{3}{8} \ S_H + \frac{1}{8} \ S_H + \ S_M = 110

final simplex tableaux
Construct the initial simplex tableau.

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

   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: Bland's rule

Find entering column \( q \) using **Bland's rule**: index of first column whose objective function coefficient is positive.

```java
private int bland()
{
    for (int q = 0; q < m + n; q++)
        if (a[m][j] > 0) return q;
    return -1;
}
```
**Simplex algorithm: min-ratio rule**

Find leaving row \( p \) using **min ratio rule**.
(Bland’s rule: if a tie, choose first such row)

<table>
<thead>
<tr>
<th>0</th>
<th>q</th>
<th>m+n</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>p</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>m</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```java
private int minRatioRule(int q) {
    int p = -1;
    for (int i = 0; i < m; i++) {
        if (a[i][q] <= 0) continue;
        else if (p == -1) p = i;
        else if (a[i][m+n] / a[i][q] < a[p][m+n] / a[p][q])
            p = i;
    }
    return p;
}
```
Simplex algorithm: pivot

**Pivot** on element row $p$, column $q$.

```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;
}
```
Execute the simplex algorithm.

```
public void solve()
{
    while (true)
    {
        int q = bland();
        if (q == -1) break;
        int p = minRatioRule(q);
        if (p == -1) ...
        
        pivot(p, q);
    }
}
```
Simplex algorithm: running time

Remarkable property. In typical practical applications, simplex algorithm terminates after at most $2(m + n)$ pivots.
- No pivot rule is known that is guaranteed to be polynomial.
- Most pivot rules are 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 rule guarantees finite # of pivots.

choose lowest valid index for entering and leaving columns
Simplex algorithm: implementation issues

To improve the bare-bones implementation.

- Avoid stalling.  
  - requires artful engineering

- Maintain sparsity.  
  - requires fancy data structures

- Numerical stability.  
  - requires advanced math

- Detect infeasibility.  
  - run "phase I" simplex algorithm

- Detect unboundedness.  
  - no leaving row

Best practice. Don't implement it yourself!

Basic implementations. Available in many programming environments.

Industrial-strength solvers. Routinely solve LPs with millions of variables.

Modeling languages. Simplify task of modeling problem as LP.
Ex 1. **OR-Objects Java library solves linear programs in Java.**

http://or-objects.org/app/library

```java
import drasys.or.mp.Problem;
import drasys.or.mp.lp.DenseSimplex;

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

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

        DenseSimplex lp = new DenseSimplex(problem);
        StdOut.println(lp.solve());
        StdOut.println(lp.getSolution());
    }
}
```
Ex 2. QSopt solves linear programs in Java or C.

http://www2.isye.gatech.edu/~wcook/qsopt

```java
import qs.*;

public class QSoptSolver {
    public static void main(String[] args) {
        Problem problem = Problem.read(args[0], false);
        problem.opt_primal();
        StdOut.println("Optimal value = " + problem.get_objval());
        StdOut.println("Optimal primal solution: ");
        problem.print_x(new Reporter(System.out), true, 6);
    }
}
```

```plaintext
% more beer.lp
Problem
   Beer
Maximize
   profit: 13A + 23B
Subject
   corn:  5A + 15B <=  480.0
   hops:  4A +  4B <=  160.0
   malt: 35A + 20B <= 1190.0
End

% java -cp .:qsopt.jar QSoptSolver beer.lp
Optimal profit = 800.0
Optimal primal solution:
   A = 12.000000
   B = 28.000000

problem in LP or MPS format
LP solvers: basic implementations

Ex 3. Microsoft Excel Solver add-in solves linear programs.
Ex 4. Matlab command `linprog` in optimization toolbox solves LPs.

```matlab
A = [5 15; 4 4; 35 20];
b = [480; 160; 1190];
c = [13; 23];
lb = [0; 0];
ub = [inf; inf];
x = linprog(-c, A, b, [], [], lb, ub)
x =
    12.0000
    28.0000
```
**LP solvers: industrial strength**

**AMPL.** [Fourer, Gay, Kernighan] An algebraic modeling language.
- Separates data from the model.
- Symbolic names for variables.
- Mathematical notation for constraints.

**CPLEX solver.** [Bixby] Highly optimized and robust industrial-strength solver.

---

```plaintext
[wayne:tombstone] ~> ampl
ILOG AMPL 9.100
AMPL Version 20021038 (SunOS 5.8)
ampl: model beer.mod;
ampl: data beer.dat;
ampl: solve;
ILOG CPLEX 9.100
CPLEX 9.1.0: optimal solution; objective 800
2 dual simplex iterations (1 in phase I)
ampl: display x;
x [*] := ale 12  beer 28 ;
```

```plaintext
% more beer.mod
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];

% more beer.dat
set PROD := beer ale;
set INGR := corn hops malt;
param: profit :=
    ale  13
    beer 23;
param: supply :=
    corn  480
    hops  160
    malt 1190;
param amt: ale beer :=
    corn  5  15
    hops  4  4
    malt 35 20;
```

---

*but license costs $$$*
• brewer’s problem
• simplex algorithm
• implementations
• duality
• modeling
**LP duality: economic interpretation**

**Brewer's problem.** Find optimal mix of beer and ale to maximize profits.

```
maximize  13A  +  23B
subject   5A  +  15B  ≤  480
          4A  +  4B  ≤  160
          35A  +  20B  ≤  1190
A, B ≥ 0
```

```
A* = 12
B* = 28
OPT = 800
```

**Entrepreneur's problem.** Buy resources from brewer to minimize cost.

- \( C, H, M = \) unit prices for corn, hops, malt.
- Brewer won't agree to sell resources if \( 5C + 4H + 35M < 13 \)
  or if \( 15C + 4H + 20M < 23 \)

```
minimize  480C  +  160H  +  1190M
subject   5C  +  4H  +  35M  ≥  13
          15C  +  4H  +  20M  ≥  23
          C, H, M ≥ 0
```

```
C* = 1
H* = 2
M* = 0
OPT = 800
```
**LP duality: sensitivity analysis**

**Q.** How much should brewer be willing to pay (marginal price) for additional supplies of scarce resources?

**A.** Corn $1, hops $2, malt $0.

**Q.** How do I compute marginal prices?

**A1.** Entrepreneur's problem is another linear program.

**A2.** Simplex algorithm solves both brewer's and entrepreneur's problems!

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

LP duality: sensitivity analysis
**LP duality theorem**

**Goal.** Given real numbers $a_{ij}$, $c_j$, $b_i$, find real numbers $x_j$ and $y_i$ that solve:

<table>
<thead>
<tr>
<th>primal problem (P)</th>
<th>dual problem (D)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>max</strong> $c_1 x_1 + c_2 x_2 + \ldots + c_n x_n$</td>
<td><strong>min</strong> $b_1 y_1 + b_2 y_2 + \ldots + b_m y_m$</td>
</tr>
<tr>
<td>subject to $a_{11} x_1 + a_{12} x_2 + \ldots + a_{1n} x_n = b_1$</td>
<td>subject to $a_{11} y_1 + a_{21} y_2 + \ldots + a_{m1} y_m = c_1$</td>
</tr>
<tr>
<td>$a_{21} x_1 + a_{22} x_2 + \ldots + a_{2n} x_n = b_2$</td>
<td>$a_{12} y_1 + a_{22} y_2 + \ldots + a_{m2} y_m = c_2$</td>
</tr>
<tr>
<td>$\vdots$ $\vdots$ $\vdots$ $\vdots$ $\vdots$ $\vdots$</td>
<td>$\vdots$ $\vdots$ $\vdots$ $\vdots$ $\vdots$ $\vdots$</td>
</tr>
<tr>
<td>$a_{m1} x_1 + a_{m2} x_2 + \ldots + a_{mn} x_n = b_m$</td>
<td>$a_{1n} y_1 + a_{2n} y_2 + \ldots + a_{mn} y_m = c_n$</td>
</tr>
<tr>
<td>$x_1, x_2, \ldots, x_n \geq 0$</td>
<td>$y_1, y_2, \ldots, y_m \geq 0$</td>
</tr>
</tbody>
</table>

**Proposition.** If (P) and (D) have feasible solutions, then max = min.
Goal. Given a matrix $A$ and vectors $b$ and $c$, find vectors $x$ and $y$ that solve:

**Primal problem (P)**

maximize $c^T x$

subject to the constraints $A x = b$ and $x \geq 0$

**Dual problem (D)**

minimize $b^T y$

subject to the constraints $A^T y \geq c$ and $y \geq 0$

Proposition. If (P) and (D) have feasible solutions, then max = min.
Brief history

1939. Production, planning. [Kantorovich]
1947. Simplex algorithm. [Dantzig]
1947. Equilibrium theory. [Koopmans]
1948. Berlin airlift. [Dantzig]
1975. Nobel Prize in Economics. [Kantorovich and Koopmans]
1979. Ellipsoid algorithm. [Khachiyan]
1990. Interior-point methods. [Nesterov-Nemirovskii, Mehota, ...]
› brewer’s problem
› simplex algorithm
› implementations
› duality
› modeling
Modeling

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**.
4. Convert to standard form.

Examples.
- Shortest paths.
- Maxflow.
- Bipartite matching.
- Assignment problem.
- 2-person zero-sum games.
...
**Maxflow problem (revisited)**

**Input.** Weighted digraph $G$, single source $s$ and single sink $t$.

**Goal.** Find maximum flow from $s$ to $t$. 

\[
\begin{align*}
\text{Maximize} & \quad x_{35} + x_{45} \\
\text{subject to the constraints} & \\
& \quad x_{01} = 2 \\
& \quad x_{02} = 2 \\
& \quad x_{13} = 1 \\
& \quad x_{14} = 1 \\
& \quad x_{23} = 1 \\
& \quad x_{24} = 1 \\
& \quad x_{35} = 2 \\
& \quad x_{45} = 2
\end{align*}
\]

Max flow from 0 to 5

<table>
<thead>
<tr>
<th></th>
<th>0-&gt;2</th>
<th>0-&gt;1</th>
<th>1-&gt;4</th>
<th>1-&gt;3</th>
<th>2-&gt;3</th>
<th>2-&gt;4</th>
<th>3-&gt;5</th>
<th>4-&gt;5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3.0</td>
<td>2.0</td>
<td>1.0</td>
<td>3.0</td>
<td>1.0</td>
<td>1.0</td>
<td>2.0</td>
<td>3.0</td>
</tr>
</tbody>
</table>
Modeling the maxflow problem as a linear program

**Variables.** \( x_{vw} = \text{flow on edge } v \rightarrow w \).

**Constraints.** Capacity and flow conservation.

**Objective function.** Net flow into \( t \).

Max flow from 0 to 5

Max flow value: 4.0

4->5 3.0 2.0
3->5 2.0 2.0
2->4 1.0 1.0
2->3 1.0 1.0
1->3 3.0 1.0
1->4 1.0 1.0
0->1 2.0 2.0
0->2 3.0 2.0

LP formulation

Maximize \( x_{35} + x_{45} \)

subject to the constraints

\[
\begin{align*}
0 & \leq x_{01} \leq 2 \\
0 & \leq x_{02} \leq 3 \\
0 & \leq x_{13} \leq 3 \\
0 & \leq x_{14} \leq 1 \\
0 & \leq x_{23} \leq 1 \\
0 & \leq x_{24} \leq 1 \\
0 & \leq x_{35} \leq 2 \\
0 & \leq x_{45} \leq 3 \\
x_{01} & = x_{13} + x_{14} \\
x_{02} & = x_{23} + x_{24} \\
x_{13} + x_{23} & = x_{35} \\
x_{14} + x_{24} & = x_{45}
\end{align*}
\]
Linear programming dual of maxflow problem

**Dual variables.** One variable $z_{vw}$ for each edge and one variable $y_v$ for each vertex.

**Dual constraints.** One inequality for each edge.

**Objective function.** Capacity of edges in cut.

\[
\begin{align*}
\text{minimize} & \quad 2z_{01} + 3z_{02} + 3z_{13} + z_{14} + z_{23} + z_{24} + 2z_{35} + 3z_{45} \\
\text{subject to the constraints} & \quad z_{01} \geq y_0 - y_1, \quad z_{23} \geq y_2 - y_3, \\
& \quad z_{02} \geq y_0 - y_2, \quad z_{24} \geq y_2 - y_4, \\
& \quad z_{13} \geq y_1 - y_3, \quad z_{35} \geq y_3 - y_5, \\
& \quad z_{14} \geq y_1 - y_4, \quad z_{45} \geq y_4 - y_5, \\
& \quad y_0 = 1, \quad y_5 = 0, \\
& \quad y_v \text{ unrestricted}, \quad z_{vw} \geq 0
\end{align*}
\]

**Interpretation.** LP dual of maxflow problem is mincut problem!
- $y_v = 1$ if $v$ is on $s$ side of min cut; $y_v = 0$ if on $t$ side.
- $z_{vw} = 1$ if $v \to w$ crosses cut.

if $y_v = 1$ and $y_w = 0$, then $z_{vw} = 1$

extreme point solution will be 0/1 (not always so lucky!)
Linear programming perspective

Q. Got an optimization problem?
Ex. Shortest paths, maxflow, matching, ... [many, many, more]

Approach 1: Use a specialized algorithm to solve it.
• Algorithms 4/e.
• Vast literature on algorithms.

Approach 2: Use linear programming.
• Many problems are easily modeled as LPs.
• Commercial solvers can solve those LPs quickly.
• Might be slower than specialized solution (but might not care).

Got an LP solver? Learn to use it!
Universal problem-solving model (in theory)

Is there a universal problem-solving model?

• Shortest paths.
• Maxflow.
• Bipartite matching.
• Assignment problem.
• Multicommodity flow.
  ...
• Two-person zero-sum games.
• Linear programming.
  ...

• Factoring
• NP-complete problems.
  ...

\[
\begin{align*}
\text{tractable} & \quad \text{intractable} \\
\{ & \\
\}
\end{align*}
\]

Does P = NP? No universal problem-solving model exists unless P = NP.