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.

\[
\begin{align*}
\text{maximize} & \quad 13A + 23B \\
\text{subject to the constraints} & \quad 5A + 15B \leq 480 \\
& \quad 4A + 4B \leq 160 \\
& \quad 35A + 20B \leq 1190 \\
& \quad A, B \geq 0
\end{align*}
\]
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.
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.
## Toy LP example: brewer’s problem

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

<table>
<thead>
<tr>
<th></th>
<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>32</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>0</td>
<td>480</td>
<td>128</td>
<td>640</td>
<td>$736</td>
</tr>
<tr>
<td>19.5</td>
<td>20.5</td>
<td>0</td>
<td>405</td>
<td>160</td>
<td>1092.5</td>
<td>$725</td>
</tr>
<tr>
<td>12</td>
<td>28</td>
<td>0</td>
<td>480</td>
<td>160</td>
<td>980</td>
<td>$800</td>
</tr>
<tr>
<td>?</td>
<td>?</td>
<td>?</td>
<td></td>
<td></td>
<td></td>
<td>&gt; $800 ?</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><strong>maximize</strong></td>
<td>$13A + 23B$</td>
<td></td>
</tr>
<tr>
<td><strong>subject to the constraints</strong></td>
<td>$5A + 15B \leq 480$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$4A + 4B \leq 160$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$35A + 20B \leq 1190$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$A, B \geq 0$</td>
<td></td>
</tr>
</tbody>
</table>
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
Regardless of objective function, optimal solution occurs at an **extreme point**.

**Brewer's problem: geometry**

Intersection of 2 constraints in 2d

![Graph showing Brewer's problem with extreme point and constraints](image)

- **(0, 0)**
- **(0, 32)**
- **(12, 28)**
- **(26, 14)**
- **(34, 0)**
Standard form linear program

**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 $x_j$.

**Caveat.** No widely agreed notion of "standard form."

matrices version

maximize $\begin{bmatrix} c_1 x_1 + c_2 x_2 + \ldots + c_n x_n \\ a_{11} x_1 + a_{12} x_2 + \ldots + a_{1n} x_n = b_1 \\ a_{21} x_1 + a_{22} x_2 + \ldots + a_{2n} x_n = b_2 \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} x_1 + a_{m2} x_2 + \ldots + a_{mn} x_n = b_m \\ x_1, x_2, \ldots, x_n \geq 0 \end{bmatrix}$

subject to the constraints

maximize $c^T x$

subject to $A x = b$

to the constraints $x \geq 0$
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>(- Z = 0)</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>
Other reductions to standard form

Minimization problem. Replace \( \min 13A + 15B \) with \( \max -13A - 15B \).

\( \geq \) constraints. Replace \( 5A + 15B \geq 480 \) with \( 5A + 15B - S_C = 480, S_C \geq 0 \).

Unrestricted variables. Replace \( A \) with \( A = A^+ - A^- \), \( A^+ \geq 0, A^- \geq 0 \).
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.
› 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 \( \Leftrightarrow \) 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</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>
Simplex algorithm: initialization

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

basis = \{ S_C, S_H, S_M \}
\[
\begin{align*}
A = B = 0 \\
Z = 0 \\
S_C = 480 \\
S_H = 160 \\
S_M = 1190
\end{align*}
\]

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 \( S_C = 480, S_H = 160, S_M = 1190 \).

one basic variable per row

no algebra needed
Simplex algorithm: pivot 1

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

which basic variable does \( B \) replace?

basis = \{ \( S_C, S_H, S_M \) \}  
\[ A = B = 0 \]  
\[ Z = 0 \]  
\[ S_C = 480 \]  
\[ S_H = 160 \]  
\[ S_M = 1190 \]

substitute \( B = (1/15) (480 - 5A - S_C) \) and add \( B \) into the basis  
(rewrite 2nd equation, eliminate \( B \) in 1st, 3rd, and 4th equations)

maximize  
\[ Z = (16/3) A - (23/15) S_C \]  
subject to the constraints  
\[ (1/3) A + B + (1/15) S_C = 32 \]  
\[ (8/3) A - (4/15) S_C + S_H = 32 \]  
\[ (85/3) A - (4/3) S_C + S_M = 550 \]  
\[ A, B, S_C, S_H, S_M \geq 0 \]

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

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

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

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)

which basic variable does A replace?

maximize \[ Z \]
subject to the constraints

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

basis = \{A, B, S_M\}
S_C = S_H = 0
Z = 800
B = 28
A = 12
S_M = 110
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: \( Z = 800 - S_C - 2S_H \)
   • Thus, optimal objective value \( Z^* \leq 800 \) since \( S_C, S_H \geq 0 \).
   • Current BFS has value 800 \( \Rightarrow \) optimal.

\[
\begin{array}{cccccc}
\text{maximize} & Z \\
\text{subject to the constraints} & \text{in table below} & & & & \\
 & \text{basis} & = & \{ A, B, S_M \} & & \\
& S_C & \geq & 0 & & \\
& S_H & \geq & 0 & & \\
& S_M & \geq & 0 & & \\
\end{array}
\]

\[
\begin{array}{cccccc}
& - & S_C & - & 2S_H & -Z = -800 \\
B & + & (1/10)S_C & + & (1/8)S_H & = 28 \\
A & - & (1/10)S_C & + & (3/8)S_H & = 12 \\
& - & (25/6)S_C & - & (85/8)S_H & +S_M & = 110 \\
A, B, S_C, S_H, S_M & \geq & 0 \\
\end{array}
\]
- brewer’s problem
- simplex algorithm
- implementations
- duality
- modeling
Simplex tableau

Encode standard form LP in a single Java 2D array.

maximize \( Z \)

\[
13A + 23B - Z = 0
\]

subject to the constraints

\[
5A + 15B + Sc = 480
\]

\[
4A + 4B + Sh = 160
\]

\[
35A + 20B + Sm = 1190
\]

\( A, B, Sc, Sh, Sm \geq 0 \)

initial simplex tableaux

\[
\begin{array}{cccccc}
5 & 15 & 1 & 0 & 0 & 480 \\
4 & 4 & 0 & 1 & 0 & 160 \\
35 & 20 & 0 & 0 & 1 & 1190 \\
13 & 23 & 0 & 0 & 0 & 0 \\
\end{array}
\]
Simplex algorithm transforms initial 2D array into solution.

maximize \( Z \)

subject to the constraints

|   | \(-\) | \(S_C\) | \(-\) | \(2S_H\) | \(-\) | \(Z\) = | -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 |
| \(A\), \(B\), \(S_C\), \(S_H\), \(S_M\) | \(\geq\) | 0 |

final simplex tableaux

\[
\begin{array}{cccccc}
0 & 1 & 1/10 & 1/8 & 0 & 28 \\
1 & 0 & -1/10 & 3/8 & 0 & 12 \\
0 & 0 & -25/6 & -85/8 & 1 & 110 \\
0 & 0 & -1 & -2 & 0 & -800 \\
m & & & & & \\
\end{array}
\]

\[
\begin{array}{ccc}
m & \leq 0 & \leq 0 \\
1 & -Z^* & \\
n & m & 1 \\
\end{array}
\]
Simplex algorithm: initial 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];
   }
```

```
          m
           A
           b

          l
           c
           0
           0

          n
           m
           1
```

- Constructor
- Put A[][] into tableau
- Put I[][] into tableau
- Put c[] into tableau
- Put b[] into tableau
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 \textbf{min ratio rule}.
(Bland’s rule: if a tie, choose first such row)

```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;
}
```
Simplex algorithm: bare-bones implementation

Execute the simplex algorithm.

```java
public void solve()
{
    while (true)
    {
        int q = bland();
        if (q == -1) break;
        int p = minRatioRule(q);
        if (p == -1) ...

        pivot(p, q);
    }
}
```

- entering column q (optimal if -1)
- leaving row p (unbounded if -1)
- pivot on row p, column 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.

---

**Smoothed Analysis of Algorithms: Why the Simplex Algorithm Usually Takes Polynomial Time**

Daniel A. Spielman*
Department of Mathematics  
M.I.T.  
Cambridge, MA 02139  
spielman@mit.edu

Shang-Hua Teng†  
Akamai Technologies Inc. and  
Department of Computer Science  
University of Illinois at Urbana-Champaign  
steng@cs.uiuc.edu
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());
   }
}
```
LP solvers: basic implementations

Ex 2. QSopt solves linear programs in Java or C.

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

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);
    }
}

% 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
Ex 3. Microsoft Excel Solver add-in solves linear programs.

LP solvers: basic implementations
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.

but license costs $$$

```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;
[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;
```
LP solvers: industrial strength

“a benchmark production planning model solved using linear programming would have taken 82 years to solve in 1988, using the computers and the linear programming algorithms of the day. Fifteen years later—in 2003—this same model could be solved in roughly 1 minute, an improvement by a factor of roughly 43 million. Of this, a factor of roughly 1,000 was due to increased processor speed, whereas a factor of roughly 43,000 was due to improvements in algorithms!”

— Designing a Digital Future

(Report to the President and Congress, 2010)

\[
\text{speedup} = \text{speedup due to big iron} \times \text{speedup due to better algorithms}
\]

43 million 1,000 43,000
• 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.

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

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 \( 15C + 4H + 20M < 23 \)

\[
\begin{align*}
\text{minimize} & \quad 480C + 160H + 1190M \\
\text{subject to} & \quad 5C + 4H + 35M \geq 13 \\
& \quad 15C + 4H + 20M \geq 23 \\
& \quad C, H + M \geq 0
\end{align*}
\]

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!

\[
\begin{array}{l}
\text{maximize} & Z \\
\text{subject to the constraints} & -SC - 2SH - Z = -800 \\
B & + (1/10) SC + (1/8) SH = 28 \\
A & - (1/10) SC + (3/8) SH = 12 \\
& - (25/6) SC - (85/8) SH + SM = 110 \\
A, B, SC, SH, SM & \geq 0
\end{array}
\]
**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></td>
<td><strong>min</strong></td>
</tr>
<tr>
<td>$c_1 x_1 + c_2 x_2 + \ldots + c_n x_n$</td>
<td>$b_1 y_1 + b_2 y_2 + \ldots + b_m y_m$</td>
</tr>
<tr>
<td>$a_{11} x_1 + a_{12} x_2 + \ldots + a_{1n} x_n = b_1$</td>
<td>$a_{11} y_1 + a_{21} y_2 + \ldots + a_{n1} y_m = c_1$</td>
</tr>
<tr>
<td>$a_{21} x_1 + a_{22} x_2 + \ldots + a_{2n} x_n = b_2$</td>
<td>$a_{12} y_1 + a_{22} y_2 + \ldots + a_{n2} y_m = c_2$</td>
</tr>
<tr>
<td>\vdots \vdots \vdots \vdots \vdots \vdots \vdots \vdots \vdots</td>
<td>\vdots \vdots \vdots \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>$a_{1n} y_1 + a_{2n} y_2 + \ldots + a_{nm} y_m = c_n$</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 $\text{max} = \text{min}$.
**LP duality theorem**

**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 $A x = b$, $x \geq 0$

**dual problem (D)**

- minimize $b^T y$
- subject to the constraints $A^T y \geq c$, $y \geq 0$

**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, Mehorta, ...]

Kantorovich  George Dantzig  von Neumann  Koopmans  Khachiyan  Karmarkar
› brewer’s problem
› simplex algorithm
› implementations
› duality
› 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.
  ...

Modeling

Modeling software usually performs this step automatically.
Maxflow problem (revisited)

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

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

Max flow from 0 to 5

0→2 3.0  2.0
0→1 2.0  2.0
1→4 1.0  1.0
1→3 3.0  1.0
2→3 1.0  1.0
2→4 1.0  1.0
3→5 2.0  2.0
4→5 3.0  3.0

Max flow value: 4.0
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 $t$.

LP formulation

Maximize $x_{35} + x_{45}$

subject to the constraints

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 $\leq x_{45} \leq 3$

$x_{01} = x_{13} + x_{14}$

$x_{02} = x_{23} + x_{24}$

$x_{13} + x_{23} = x_{35}$

$x_{14} + x_{24} = x_{45}$

Max flow value: 4.0

4 $\rightarrow$ 5 3.0 2.0
3 $\rightarrow$ 5 2.0 2.0
2 $\rightarrow$ 4 1.0 1.0
2 $\rightarrow$ 3 1.0 1.0
1 $\rightarrow$ 3 3.0 1.0
1 $\rightarrow$ 4 1.0 1.0
0 $\rightarrow$ 1 2.0 2.0
0 $\rightarrow$ 2 3.0 2.0

Max flow from 0 to 5
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.

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 \(v \rightarrow w\) crosses cut.
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!
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.
  ...

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