Linear programming

What is it? 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.

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.
- **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.

**Brewer’s problem: feasible region**

Inequalities define halfplanes; feasible region is a convex polygon.

**Brewer’s problem: linear programming formulation**

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

**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>0</td>
<td>179</td>
<td>136</td>
<td>1190</td>
<td>$442</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>32</td>
<td>480</td>
<td>128</td>
<td>640</td>
<td>$736</td>
<td></td>
</tr>
<tr>
<td>19.5</td>
<td>20.5</td>
<td>405</td>
<td>160</td>
<td>1092.5</td>
<td>$725</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>28</td>
<td>480</td>
<td>160</td>
<td>980</td>
<td>$800</td>
<td></td>
</tr>
<tr>
<td>?</td>
<td>?</td>
<td>34</td>
<td>160</td>
<td>1190</td>
<td>&gt; $800</td>
<td></td>
</tr>
</tbody>
</table>

$5A + 15B \leq 480$

$4A + 4B \leq 160$

$35A + 20B \leq 1190$

$A, B \geq 0$

**Toy LP example: brewer’s problem**

Goods are divisible

$34 \text{ barrels } \times 35 \text{ lbs malt } = 1190 \text{ lbs}$

[amount of available malt]
Brewer’s problem: objective function

**Goal.** Maximize linear objective function of \( n \) nonnegative variables, subject to \( m \) linear equations.
- Input: real numbers \( a_{ij}, c_i, b_i \).
- Output: real numbers \( x_j \).

<table>
<thead>
<tr>
<th>primal problem (P)</th>
<th>matrix version</th>
</tr>
</thead>
<tbody>
<tr>
<td>maximize ( c_1 x_1 + c_2 x_2 + \ldots + c_n x_n )</td>
<td>maximize ( c' x )</td>
</tr>
<tr>
<td>subject to the constraints</td>
<td>subject to ( A x = b )</td>
</tr>
<tr>
<td>( a_{11} x_1 + a_{12} x_2 + \ldots + a_{1n} x_n = b_1 )</td>
<td>to the constraints ( x \geq 0 )</td>
</tr>
<tr>
<td>( a_{21} x_1 + a_{22} x_2 + \ldots + a_{2n} x_n = b_2 )</td>
<td></td>
</tr>
<tr>
<td>( \vdots )</td>
<td>( \vdots )</td>
</tr>
<tr>
<td>( a_{m1} x_1 + a_{m2} x_2 + \ldots + a_{mn} x_n = b_m )</td>
<td></td>
</tr>
<tr>
<td>( x_1, x_2, \ldots, x_n \geq 0 )</td>
<td></td>
</tr>
</tbody>
</table>

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

Brewer’s problem: geometry

Optimal solution occurs at an **extreme point.**

Intersection of 2 constraints in 2d

Standard form linear program

**Original formulation.**

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

**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.

**Converting the brewer’s problem to the standard form**

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

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.

- Good news: number of extreme points to consider is finite.
- Bad news: number of extreme points can be exponential!

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

Linear Programming

- brewer's problem
- simplex algorithm
- implementations
- reductions
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: initialization

\[
\begin{align*}
\text{maximize} & \quad Z = 13A + 23B \\
\text{subject to the constraints} & \quad 5A + 15B + Sc = 480 \\
& \quad 4A + 4B + Sh = 160 \\
& \quad 35A + 20B + Sm = 1190 \\
\end{align*}
\]

Initial basic feasible solution.

- Start with slack variables \( \{ Sc, Sh, Sm \} \) as the basis.
- Set non-basic variables \( A \) and \( B \) to 0.
- 3 equations in 3 unknowns yields \( Sc = 480, Sh = 160, Sm = 1190 \).

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 \( \iff \) BFS.
- BFS \( \iff \) extreme point.

Substitute \( B = (1/15) (480 - 5A - Sc) \) and add \( B \) into the basis.

Simplex algorithm: pivot 1

\[
\begin{align*}
\text{maximize} & \quad Z = 13A + 23B \\
\text{subject to the constraints} & \quad 5A + 15B + Sc = 480 \\
& \quad 4A + 4B + Sh = 160 \\
& \quad 35A + 20B + Sm = 1190 \\
\end{align*}
\]

which basic variable does \( B \) replace?

\[
\begin{align*}
\text{maximize} & \quad Z = 16/3 A - 23/15 Sc \\
\text{subject to the constraints} & \quad (1/3) A + B + (1/15) Sc = 32 \\
& \quad (8/3) A - (4/3) Sc + Sh = 32 \\
& \quad (85/3) A - (4/3) Sc + Sm = 550 \\
\end{align*}
\]

no algebra needed
Simplex algorithm: pivot 1

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: $\min \{480/15, 160/4, 1190/20\}$.

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.
LINEAR PROGRAMMING

- brewer's problem
- simplex algorithm
- implementations
- reductions

LINEAR PROGRAMMING

Simplex tableau

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

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

Simplex tableau

Initial simplex tableaux

```
| 5 15 1 0 0 | 480 |
| 4 4 0 1 0 | 160 |
| 35 20 0 0 1 | 1190 |
| 13 23 0 0 0 | 0 |
```

Simplex algorithm: initial simplex tableaux

Construct the initial simplex tableau.

```
public class Simplex {
    private double[][] a; // simplex tableau
    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] = 1.0;
        for (int i = 0; i < m; i++)
            for (int j = i; j < m; j++)
                a[i][m+j] = c[j];
        for (int i = 0; i < m; i++)
            for (int j = 0; j < m; j++)
                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.

Simplex algorithm: min-ratio rule

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

Simplex algorithm: pivot

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

Simplex algorithm: bare-bones implementation

Execute the simplex algorithm.
Remarkable property. In typical practical applications, simplex algorithm terminates after at most $2(m + n)$ pivots.

"Yes. Most of the time it solved problems with $m$ equations in $2m$ or $3m$ steps—that was truly amazing. I certainly did not anticipate that it would turn out to be so terrific. I had had no experience at the time with problems in higher dimensions, and I didn’t trust my geometrical intuition. For example, my intuition told me that the procedure would require too many steps wandering from one adjacent vertex to the next. In practice it takes few steps. In brief, one’s intuition in higher dimensional space is not worth a damn! Only now, almost forty years from the time when the simplex method was first proposed, are people beginning to get some insight into why it works as well as it does."

— George Dantzig 1984

Degeneracy. New basis, same extreme point.

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.

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

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

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

\( \geq \) constraints. Replace \( 4A + 4B \geq 160 \) with \( 4A + 4B - S_H = 160, S_H \geq 0 \).

Unrestricted variables. Replace \( B \) with \( B = B_0 - B_1 \), \( B_0 \geq 0, B_1 \geq 0 \).

Maxflow problem (revisited)

**Input.** Weighted digraph \( G \), single source \( s \) and single sink \( t \).

**Goal.** Find maximum flow from \( s \) to \( t \).

Modeling the maxflow problem as a linear program

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

**Constraints.** Capacity and flow conservation.

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

Linear “programming” (1950s term) = reduction to LP (modern term).

1. Identify variables.
2. Define constraints (inequalities and equations).
3. Define objective function.
4. Convert to standard form. **software usually performs this step automatically**

Examples.

- Maxflow.
- Shortest paths.
- Bipartite matching.
- Assignment problem.
- 2-person zero-sum games.
- ...
Maximum cardinality bipartite matching problem

**Input.** Bipartite graph.

**Goal.** Find a matching of maximum cardinality.

**Interpretation.** Mutual preference constraints.
- People to jobs.
- Students to writing seminars.

<table>
<thead>
<tr>
<th>Alice</th>
<th>Adobe, Apple, Google</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bob</td>
<td>Adobe, Apple, Google</td>
</tr>
<tr>
<td>Carol</td>
<td>Google, IBM, Sun</td>
</tr>
<tr>
<td>Dave</td>
<td>Adobe, Apple, Google</td>
</tr>
<tr>
<td>Eliza</td>
<td>IBM, Sun, Yahoo</td>
</tr>
<tr>
<td>Frank</td>
<td>Google, Sun, Yahoo</td>
</tr>
</tbody>
</table>

Example: job offers

Matching of cardinality 6: A–1, B–5, C–2, D–0, E–3, F–4

Linear programming perspective

**Q.** Got an optimization problem?

**Ex.** Maxflow, bipartite matching, shortest paths, ... [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.
- Might be slower than specialized solution (but you might not care).

Got an LP solver? Learn to use it!

Universal problem-solving model (in theory)

**Is there a universal problem-solving model?**
- Maxflow.
- Shortest paths.
- 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.

Maximum cardinality bipartite matching problem

**LP formulation.** One variable per pair.

**Interpretation.** $x_{ij} = 1$ if person $i$ assigned to job $j$.

\[
\begin{align*}
\text{maximize} & \quad x_{10} + x_{11} + x_{12} + x_{20} + x_{21} + x_{22} + x_{32} + x_{33} + x_{43} + x_{44} + x_{54} + x_{55} \\
\text{subject to the constraints} & \quad \begin{aligned}
& x_{10} + x_{11} + x_{12} \leq 1 \\
& x_{20} + x_{21} + x_{22} \leq 1 \\
& x_{32} + x_{33} + x_{34} \leq 1 \\
& x_{43} + x_{44} + x_{45} \leq 1 \\
& x_{54} + x_{55} \leq 1
\end{aligned}
\end{align*}
\]

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

**Corollary.** Can solve matching problem by solving LP.

See next lecture
LINEAR PROGRAMMING

- brewer's problem
- simplex algorithm
- implementations
- reductions