LINEAR PROGRAMMING

- brewer’s problem
- simplex algorithm
- implementations
- reductions
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, ...

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

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.
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.
LINEAR PROGRAMMING

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

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.
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>34 barrels × 35 lbs malt = 1190 lbs [amount of available malt]</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Goods are divisible: corn (480 lbs), hops (160 oz), malt (1190 lbs)

- Ale profit: $13 per barrel
- Beer profit: $23 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.

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

Inequalities define halfplanes; feasible region is a convex polygon.
Brewer’s problem: objective function

13A + 23B = $800
13A + 23B = $1600
13A + 23B = $442
Brewer’s problem: geometry

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

intersection of 2 constraints in 2d

![Diagram showing the optimal solution for Brewer's problem with a shaded feasible region and extreme points at (0, 32), (12, 28), and (26, 14).]
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."
Converting the brewer’s problem to the standard form

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.

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

- brewer’s problem
- simplex algorithm
- implementations
- reductions
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 20\textsuperscript{th} 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 $\implies$ BFS.
- BFS $\iff$ extreme point.

<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>$5A + 15B + S_C = 480$</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>$A, B, S_C, S_H, S_M \geq 0$</td>
<td></td>
</tr>
</tbody>
</table>
Simplex algorithm: initialization

maximize

\[
\begin{array}{ccc}
\text{Z} & = & 0 \\
13A & + & 23B \\
5A & + & 15B & + S_C \\
4A & + & 4B & + S_H \\
35A & + & 20B & + S_M \\
A, & B, & S_C, & S_H, & S_M & \geq & 0
\end{array}
\]

subject to the constraints

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

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**

**maximize** \[ Z \]

<table>
<thead>
<tr>
<th>( 13A + 23B - Z )</th>
<th>pivot</th>
</tr>
</thead>
<tbody>
<tr>
<td>13A + 23B - Z = 0</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_H , S_M ( \geq 0 )</td>
<td></td>
</tr>
</tbody>
</table>

**subject to the constraints**


**substitute** \( B = (1/15) (480 - 5A - S_C) \) and add B into the basis

(number 2nd equation, eliminate B in 1st, 3rd, and 4th equations)

**basis** = \( \{ S_C , S_H , S_M \} \)

- \( A = B = 0 \)
- \( Z = 0 \)
- \( S_C = 480 \)
- \( S_H = 160 \)
- \( S_M = 1190 \)

which basic variable does B replace?

**maximize** \[ Z \]

<table>
<thead>
<tr>
<th>( (16/3)A - (23/15)S_C - Z )</th>
<th>pivot</th>
</tr>
</thead>
<tbody>
<tr>
<td>(16/3)A - (23/15)S_C - Z = -736</td>
<td></td>
</tr>
<tr>
<td>(1/3)A + B + (1/15)S_C = 32</td>
<td></td>
</tr>
<tr>
<td>(8/3)A - (4/15)S_C + S_H = 32</td>
<td></td>
</tr>
<tr>
<td>(85/3)A - (4/3)S_C + S_M = 550</td>
<td></td>
</tr>
<tr>
<td>A , B , S_C , S_H , S_M ( \geq 0 )</td>
<td></td>
</tr>
</tbody>
</table>

**basis** = \( \{ B , S_H , S_M \} \)

- \( A = S_C = 0 \)
- \( Z = 736 \)
- \( B = 32 \)
- \( S_H = 32 \)
- \( S_M = 550 \)
Simplex algorithm: pivot 1

maximize

\[
\begin{array}{cccc}
Z & 13A & + & 23B & - & Z = 0 \\
5A & + & 15B & + & S_c & = 480 \\
4A & + & 4B & + & S_h & = 160 \\
35A & + & 20B & + & S_m & = 1190 \\
A, B, S_c, S_h, S_m & \geq & 0 \\
\end{array}
\]

basis = \{S_c, S_h, S_m\}

\[
A = B = 0 \\
Z = 0 \\
S_c = 480 \\
S_h = 160 \\
S_m = 1190
\]

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: pivot 2

maximize \( Z \)

subject to the constraints

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

basis = \{ B, S_H, S_M \}
\begin{align*}
A &= S_C = 0 \\
Z &= 736 \\
B &= 32 \\
S_H &= 32 \\
S_M &= 550
\end{align*}

which basic variable does \( A \) replace?

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

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

basis = \{ A, B, S_M \}
\begin{align*}
S_C &= S_H = 0 \\
Z &= 800 \\
B &= 28 \\
A &= 12 \\
S_M &= 110
\end{align*}
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 ⇒ optimal.

| maximize \( Z \) | \( \begin{array}{cccc}
- & S_C & - & 2 S_H & - Z = -800 \\
subject to the constraints & 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 \\
basis = \{ A, B, S_M \} \\
S_C = S_H = 0 \\
Z = 800 \\
B = 28 \\
A = 12 \\
S_M = 110 |
LINEAR PROGRAMMING

- brewer’s problem
- simplex algorithm
- implementations
- reductions
LINEAR PROGRAMMING

- brewer’s problem
- simplex algorithm
- implementations
- reductions
Simplex tableau

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>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>15</td>
<td>1</td>
<td>0</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>0</td>
<td>160</td>
</tr>
<tr>
<td>35</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1190</td>
</tr>
<tr>
<td>13</td>
<td>23</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

initial simplex tableaux
Simplex tableau

Simplex algorithm transforms initial 2D array into solution.

maximize $Z$

subject to the constraints

$- S_C - 2 S_H - Z = -800$

$B + (1/10) S_C + (1/8) S_H = 28$

$A - (1/10) S_C + (3/8) S_H = 12$

$- (25/6) S_C - (85/8) S_H + S_M = 110$

$A, B, S_C, S_H, S_M \geq 0$

Final simplex tableaux

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

$x^*$

$\leq 0 \quad \leq 0 \quad -Z^*$
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];
    }
```

```
<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>l</td>
<td>c</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>n</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>m</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>l</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
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][q] > 0) return q;
    return -1;
}
```

- entering column $q$ has positive objective function coefficient
- optimal
Simplex algorithm: min-ratio rule

Find leaving row $p$ using **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;
}
```

- Scale all entries but row \( p \) and column \( q \)
- Zero out column \( q \)
- Scale row \( p \)
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
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
Remarkable property. In typical practical applications, simplex algorithm terminates after at most $2(m + n)$ 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.

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.
“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)
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, ...]
LINEAR PROGRAMMING

- brewer’s problem
- simplex algorithm
- implementations
- reductions
LINEAR PROGRAMMING

- brewer’s problem
- simplex algorithm
- implementations
- reductions
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 \( B = B_0 - B_1, \ B_0 \geq 0, \ B_1 \geq 0 \).

\[
\begin{align*}
\text{nonstandard form} \\
& \text{minimize} \quad 13A + 15B \\
& \text{subject to:} \\
& \quad 5A + 15B \leq 480 \\
& \quad 4A + 4B \geq 160 \\
& \quad 35A + 20B = 1190 \\
& \quad A \geq 0 \\
& \quad B \text{ is unrestricted}
\end{align*}
\]

\[
\begin{align*}
\text{standard form} \\
& \text{maximize} \quad -13A - 15B_0 + 15B_1 \\
& \text{subject to:} \\
& \quad 5A + 15B_0 - 15B_1 + S_C = 480 \\
& \quad 4A + 4B_0 - 4B_1 - S_H = 160 \\
& \quad 35A + 20B_0 - 20B_1 = 1190 \\
& \quad A, B_0, B_1, S_C, S_H \geq 0
\end{align*}
\]
Modeling

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

- 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.  
   software usually performs this step automatically

Examples.

- Maxflow.
- Shortest paths.
- 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 $t$.

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 2.0
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

- \( 0 \leq x_{01} \leq 2 \)
- \( 0 \leq x_{02} \leq 3 \)
- \( 0 \leq x_{13} \leq 3 \)
- \( 0 \leq x_{14} \leq 1 \)
- \( 0 \leq x_{23} \leq 1 \)
- \( 0 \leq x_{24} \leq 1 \)
- \( 0 \leq x_{35} \leq 2 \)
- \( 0 \leq x_{45} \leq 3 \)

\[
\begin{align*}
0 &= x_{01} + x_{13} + x_{14} \\
0 &= x_{02} + x_{23} + x_{24} \\
x_{13} + x_{23} &= x_{35} \\
x_{14} + x_{24} &= x_{45}
\end{align*}
\]
Maximum cardinality bipartite matching problem

**Input.** Bipartite graph.

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

set of edges with no vertex appearing twice

**Interpretation.** Mutual preference constraints.

- People to jobs.
- Students to writing seminars.

**Example: job offers**

| Alice       | Adobe Apple, Google |
| Bob         | Adobe, Apple, Yahoo |
| Carol       | Google, IBM, Sun    |
| Dave        | Adobe, Apple, Apple |
| Eliza       | IBM, Sun, Yahoo     |
| Frank       | Google, Sun, Yahoo  |
|             | Adobe               |
|             | Alice, Bob, Dave    |
|             | Apple               |
|             | Alice, Bob, Dave    |
|             | Google              |
|             | Alice, Carol, Frank |
|             | IBM                 |
|             | Carol, Eliza        |
|             | Sun                 |
|             | Carol, Eliza, Frank |
|             | Yahoo               |
|             | Bob, Eliza, Frank   |

matching of cardinality 6: A–1, B–5, C–2, D–0, E–3, F–4
Maximum cardinality bipartite matching problem

LP formulation. One variable per pair.
Interpretation. \( x_{ij} = 1 \) if person \( i \) assigned to job \( j \).

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. not usually so lucky!
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!
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.

Universal problem-solving model (in theory)

isi 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.
LINEAR PROGRAMMING

- brewer’s problem
- simplex algorithm
- implementations
- reductions
LINEAR PROGRAMMING

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