6. Linear Programming

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, ...

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

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.
- **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.
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: 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></td>
<td>&gt; $800</td>
</tr>
</tbody>
</table>

Linear programming formulation.
- Let \( A \) be the number of barrels of ale.
- Let \( B \) be the number of barrels of beer.

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

### Linear programming formulation

- Let \( A \) be the number of barrels of ale.
- Let \( B \) be the number of barrels of beer.

\[
\begin{align*}
\text{maximize} & \quad 13A + 23B \\
\text{subject to} & \quad 5A + 15B \leq 480 \\
& \quad 4A + 4B \leq 160 \\
& \quad 35A + 20B \leq 1190 \\
& \quad A, B \geq 0
\end{align*}
\]
Inequalities define halfplanes; feasible region is a convex polygon.

Brewer's problem: feasible region
(34, 0)  
(0, 32)  
(12, 28)  
(26, 14)  
(0, 0)  

ale  
beer  
corn

5A + 15B ≤ 480  
4A + 4B ≤ 160  
35A + 20B ≤ 1190

Regardless of objective function, optimal solution occurs at an extreme point.

Brewer's problem: geometry
intersection of 2 constraints in 2d

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 $x_i$.

primal problem (P)

<table>
<thead>
<tr>
<th>maximize</th>
<th>$c_1 x_1 + c_2 x_2 + \ldots + c_n x_n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>subject to the constraints</td>
<td>$a_{11} x_1 + a_{12} x_2 + \ldots + a_{1n} x_n = b_1$</td>
</tr>
<tr>
<td></td>
<td>$\vdots$</td>
</tr>
<tr>
<td></td>
<td>$a_{m1} x_1 + a_{m2} x_2 + \ldots + a_{mn} x_n = b_m$</td>
</tr>
<tr>
<td>$x_1, x_2, \ldots, x_n \geq 0$</td>
<td></td>
</tr>
</tbody>
</table>

matrix version

maximize $c^T x$
subject to the constraints $A x = b$
$x \geq 0$
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.

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

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.

- 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.
Simplex algorithm

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.

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.

Simplex algorithm: initialization

<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>$5A + 15B + S_c$</td>
<td>$= 480$</td>
</tr>
<tr>
<td>$4A + 4B + S_H$</td>
<td>$= 160$</td>
</tr>
<tr>
<td>$35A + 20B + S_M$</td>
<td>$= 1190$</td>
</tr>
<tr>
<td>$A, B, S_c, S_H, S_M \geq 0$</td>
<td></td>
</tr>
</tbody>
</table>

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 to the constraints</td>
<td>$13A + 23B - Z = 0$</td>
</tr>
<tr>
<td>$5A + 15B + S_c$</td>
<td>$= 480$</td>
</tr>
<tr>
<td>$4A + 4B + S_H$</td>
<td>$= 160$</td>
</tr>
<tr>
<td>$35A + 20B + S_M$</td>
<td>$= 1190$</td>
</tr>
<tr>
<td>$A, B, S_c, S_H, S_M \geq 0$</td>
<td></td>
</tr>
</tbody>
</table>

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)

<table>
<thead>
<tr>
<th>maximize</th>
<th>$Z$</th>
</tr>
</thead>
<tbody>
<tr>
<td>subject to the constraints</td>
<td>$(16/3) A - (23/15) S_c - Z = -736$</td>
</tr>
<tr>
<td>$(1/3) A + B + (1/15) S_c$</td>
<td>$= 32$</td>
</tr>
<tr>
<td>$(8/3) A - (4/3) S_c + S_H$</td>
<td>$= 32$</td>
</tr>
<tr>
<td>$(85/3) A - (4/3) S_c + S_M$</td>
<td>$= 550$</td>
</tr>
<tr>
<td>$A, B, S_c, S_H, S_M \geq 0$</td>
<td></td>
</tr>
</tbody>
</table>
Simplex algorithm: pivot 1

maximize \[ Z = 13A + 23B \]
subject to the constraints
\[
\begin{align*}
5A + 15B + Sc & = 480 \\
4A + 4B + Sh & = 160 \\
35A + 20B + SM & = 1190 \\
A, B, Sc, Sh, SM & \geq 0
\end{align*}
\]

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: $Z = 800 - Sc - 2 Sh$
  • Thus, optimal objective value $Z^* \leq 800$ since $Sc, Sh \geq 0$.
  • Current BFS has value $800 \Rightarrow$ optimal.

Simplex algorithm: pivot 2

maximize \[ Z = (16/3) A - (23/15) Sc - 160 \]
subject to the constraints
\[
\begin{align*}
(1/3) A + B + (1/15) Sc & = 32 \\
(8/3) A - (4/5) Sc + Sh & = 32 \\
(85/3) A - (4/3) Sc + Sm & = 550 \\
A, B, Sc, Sh, Sm & \geq 0
\end{align*}
\]

Substitute $A = (3/8) (32 + (4/15) Sc - Sh)$ and add $A$ into the basis (rewrite 3rd equation, eliminate $A$ in 1st, 2nd, and 4th equations)

maximize \[ Z = -800 \]
subject to the constraints
\[
\begin{align*}
B + (1/10) Sc + (1/8) Sh & = 28 \\
A - (1/10) Sc + (3/8) Sh & = 12 \\
- (25/6) Sc - (85/8) Sh + Sm & = 110 \\
A, B, Sc, Sh, Sm & \geq 0
\end{align*}
\]
Encode standard form LP in a single Java 2D array.

Simplex tableau

maximize $Z$
subject to the constraints

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

| \(5\) | \(15\) | \(1\) | \(0\) | \(0\) | \(480\) |
| \(4\) | \(4\) | \(0\) | \(1\) | \(0\) | \(160\) |
| \(35\) | \(20\) | \(0\) | \(0\) | \(1\) | \(1190\) |
| \(13\) | \(23\) | \(0\) | \(0\) | \(0\) | \(0\) |

Initial simplex tableaux

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];
   }

   \(\text{construct}\)

   put \(a[i][j]\) into tableau
   put \(z[i][j]\) into tableau
   put \(e[i]\) into tableau
   put \(a[j]\) into tableau
}

Simplex simplex algorithm

Simplex tableau

maximize $Z$
subject to the constraints

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

| \(0\) | \(1\) | \(1/10\) | \(1/8\) | \(0\) | \(28\) |
| \(1\) | \(0\) | \(-1/10\) | \(3/8\) | \(0\) | \(12\) |
| \(0\) | \(0\) | \(-25/6\) | \(-85/8\) | \(1\) | \(110\) |
| \(0\) | \(0\) | \(-1\) | \(-2\) | \(0\) | \(-800\) |

Final simplex tableaux

Simplex algorithm: Bland’s rule

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

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)

```
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 \( q \).

```
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) a[i][j] -= a[i][q] * a[p][j] / a[p][q];
   a[p][q] = 1.0;
}
```

Simplex algorithm: bare-bones implementation

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);
   }
}
```

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
teng@cs.uiuc.edu
Simplex algorithm: degeneracy

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

“stalling” is common in practice

![Graph showing degeneracy](image)

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.

Ex 1. OR-Objects Java library solves linear programs in Java.

```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.setRightHandSide("corn", 5.0);
      problem.setCoefficientAt("corn", "x1", 15.0);
      problem.setCoefficientAt("hops", "x1", 4.0);
      problem.setCoefficientAt("hops", "x2", 4.0);
      problem.setCoefficientAt("malt", "x1", 25.0);
      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.

```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);
   }
}
```
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)
```

```matlab
x =
    12.0000
    28.0000
```

LP solvers: basic implementations

- Sorry, no longer support on Mac.

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.

```
[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

but license costs $$$
```
**LP duality: economic interpretation**

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

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

- **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 to the constraints
  5C + 4H + 35M ≥ 13
  15C + 4H + 20M ≥ 23
  C , H + M ≥ 0
```

**LP duality theorem**

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

**primal problem (P)**

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

**dual problem (D)**

```
minimize $b_1 y_1 + b_2 y_2 + ... + b_m y_m$
subject to
  $a_{11} y_1 + a_{12} y_2 + ... + a_{1m} y_m = c_1$
  $a_{21} y_1 + a_{22} y_2 + ... + a_{2m} y_m = c_2$
  ... ...
  $a_{m1} y_1 + a_{m2} y_2 + ... + a_{mn} y_m = c_n$
  $y_1 , y_2 , ..., y_m ≥ 0$
```

**Proposition.** If (P) and (D) have feasible solutions, then $\max = \min$.

**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
  $A x = b$
  $x ≥ 0$
```

**dual problem (D)**

```
minimize $b^T y$
subject to
  $A^T y ≥ c$
  $y ≥ 0$
```

**Proposition.** If (P) and (D) have feasible solutions, then $\max = \min$. 

**LP duality theorem**

**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!
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, ...]

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 $t$.

Example of reducing network
Modeling the maxflow problem as a linear program

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

**Constraints.** Capacity and flow conservation.

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

\[
\text{Maximize } x_{v5} + x_{45}
\]

subject to the constraints

\[
\begin{align*}
0 & \leq x_{01} \leq 2 \\
0 & \leq x_{02} \leq 3 \\
0 & \leq x_{12} \leq 3 \\
0 & \leq x_{13} \leq 1 \\
0 & \leq x_{14} \leq 1 \\
0 & \leq x_{15} \leq 2 \\
0 & \leq x_{23} \leq 3 \\
x_{01} &= x_{13} + x_{14} \\
x_{02} &= x_{23} + x_{24} \\
x_{13} + x_{23} &= x_{35} \\
x_{14} + x_{24} &= x_{45}
\end{align*}
\]

Linear programming 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!**

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.

\[
\begin{align*}
\text{minimize} & \quad 2z_{01} + 3z_{02} + 3z_{13} + z_{14} + z_{23} + z_{24} + 2z_{35} + 3z_{45} \\
\text{subject to} & \quad z_{01} \geq y_v - y_t \\
& \quad z_{02} \geq y_v - y_t \\
& \quad z_{13} \geq y_v - y_t \\
& \quad z_{14} \geq y_v - y_t \\
& \quad z_{23} \geq y_v - y_t \\
& \quad z_{24} \geq y_v - y_t \\
& \quad z_{35} \geq y_v - y_t \\
& \quad y_v = 1 \\
\end{align*}
\]

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

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.
- ...

**Does P = NP?** No universal problem-solving model exists unless \( P = NP \).

**Factoring**

- \( y_v = 1 \) and \( y_w = 0 \), then \( z_{vw} = 1 \)

**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.

**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.
- ...

**Does P = NP?** No universal problem-solving model exists unless \( P = NP \).