“surrounded by notational and terminological thickets”

“both of these defenses are lovingly cultivated by a coterie of stern acolytes who have devoted themselves to the field…”

Numerical Recipes (Press, et. al)
Linear programming

Mathematical formulation of optimization problems

next (last?) step towards UNIVERSAL PROBLEM-SOLVING MODEL
  • for combinatorial optimization problems

Like mincost flow, LP is important for two primary reasons

it is a GENERAL PROBLEM-SOLVING MODEL
  • solves (through reduction) numerous practical problems
  • more general than mincost flow

it is TRACTABLE and PRACTICAL
  • we know fast algorithms that solve LP problems
  • more complicated than mincost-flow algorithms

SIMPLEX method generalizes network simplex algorithm
Reduction

**SOLUTION** (algorithm designer’s goal)
- an **EFFICIENT** algorithm that can ALWAYS find the answer

*Ex:* network simplex algorithms solve mincost flow

**REDUCTION** (algorithm designer’s main tool)
- solve a problem by converting it to another

*Ex:* Bipartite matching reduces to maxflow

Reduction, in practice,
- may provide quick solution
- is better than brute force
- is better than investing years of research to understand details of a problem
- gives insight into difficulty

Vast array of tractable problems reduce to LP
Simplex algorithm solves practical instances efficiently
Linear programming

Maximize OBJECTIVE FUNCTION subject to CONSTRAINTS

Ex:

• maximize \( x+y \)
• subject to the constraints
  \[-x + y \leq 5\]
  \[x + 4y \leq 45\]
  \[2x + y \leq 27\]
  \[3x - 4y \leq 24\]
  \[x, y \geq 0\]

Vast number of applications

"LINEAR": no \( x^2, xy, \text{ etc.} \)

"PROGRAMMING": formulate the equations
  (reduce given problem to linear programming)
Given table of food nutrients and costs

<table>
<thead>
<tr>
<th></th>
<th>bacon</th>
<th>eggs</th>
<th>cereal</th>
<th>milk</th>
<th>MDR</th>
</tr>
</thead>
<tbody>
<tr>
<td>protein</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>vitamin A</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>iron</td>
<td>2</td>
<td>5</td>
<td>0</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>$</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

Find the least expensive breakfast that provides the minimum daily requirements

LP formulation

- minimize $2b + e + c + 2m$
- subject to the constraints
  
  $3b + 2e \geq 1$
  $5b + e + c \geq 3$
  $2b + 5e + m \geq 4$
  $b, e, c, m \geq 0$
LP example: mincost flow reduction

- maximize $c_3$
- subject to the constraints
  
  $e_{01} \leq 2$
  $e_{02} \leq 3$
  $e_{13} \leq 3$
  $e_{14} \leq 1$
  $e_{23} \leq 1$
  $e_{24} \leq 1$
  $e_{35} \leq 2$
  $e_{45} \leq 3$
  $e_{50} = e_{01} + e_{02}$
  $e_{01} = e_{13} + e_{14}$
  $e_{02} = e_{23} + e_{24}$
  $e_{35} = e_{13} + e_{23}$
  $e_{45} = e_{14} + e_{24}$
  $e_{50} = e_{35} + e_{45}$
  $c_1 = 3e_{01} + e_{02} + e_{13} + e_{14}$
  $c_2 = 4e_{23} + 2e_{24} + 2e_{35} + e_{45}$
  $c_3 = 9e_{50} - c_1 - c_2$
  $\forall \geq 0$
Geometric interpretation (two variables)

Inequalities: halfplanes
intersecting halfplanes: convex polygon

- maximize \( x + y \)
- subject to the constraints
  - \(-x + y \leq 5\)
  - \(x + 4y \leq 45\)
  - \(2x + y \leq 27\)
  - \(3x - 4y \leq 24\)
  - \(x, y \geq 0\)

**THM** Objective function is maximized at a simplex

**INFEASIBLE:** add \( x > 13\)

**REDUNDANT:** add \( x < 13\)

**UNBOUNDED:** delete 2nd and 3rd inequalities
Geometric interpretation (three variables)

Inequalities: halfspaces
intersecting halfspaces: convex polytope

Higher dimensions: multifaceted convex polytope (SIMPLEX)
- \( N \) inequalities in \( k \) vars could have \((k-1)^N\) simplex vertices
Standard form of LPs

- maximize $x+y+z$
- subject to the constraints
  - $-x + y \leq 5$
  - $x + 4y \leq 45$
  - $2x + y \leq 27$
  - $3x - 4y \leq 24$
  - $z \leq 4$
  - $x, y, z \geq 0$

Add SLACK variables to convert inequalities to equations
- constrain all variables to be nonnegative

- $-x + y + a = 5$
- $x + 4y + b = 45$
- $2x + y + c = 27$
- $3x - 4y + d = 24$
- $z + e = 4$

M SIMULTANEOUS EQUATIONS with excess variables
choose $M$ of the variables as a BASIS

- solve by setting nonbasis variables to 0
- corresponds to a simplex vertex

$$
\begin{array}{cccccccccc}
. & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
. & -1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 5 \\
. & 1 & 4 & 0 & 0 & 1 & 0 & 0 & 0 & 45 \\
. & 2 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 27 \\
. & 3 & -4 & 0 & 0 & 0 & 0 & 1 & 0 & 24 \\
. & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 4 \\
\end{array}
$$

PIVOT STEP: add one variable to the basis, drop another

- add appropriate multiple of row to every other row

$$
\begin{array}{cccccccccc}
. & 0 & -7 & -3 & 0 & 0 & 0 & 1 & 0 & 24 \\
. & 0 & -1 & 0 & 1 & 0 & 0 & 1 & 0 & 39 \\
. & 0 & 16 & 0 & 0 & 1 & 0 & -1 & 0 & 111 \\
. & 0 & 8 & 0 & 0 & 0 & 1 & -2 & 0 & 33 \\
. & 1 & -4 & 0 & 0 & 0 & 0 & 1 & 0 & 24 \\
. & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 4 \\
\end{array}
$$

corresponds to moving to an adjacent simplex vertex
**Simplex algorithm**

**THM:** Any LP problem is either unbounded or maximized at a simplex vertex. The simplex algorithm finds such

**Generic simplex algorithm**
- start at some simplex vertex
- increase objective function by pivoting to move to an adjacent vertex

**Issues**
- which adjacent vertex?
- avoid degenerate pivots (new basis, same vertex)
- avoid cycles (get stuck on a set of vertices)?

**Code not much more complicated than Gaussian elimination**
- [admittedly, many details omitted!]
Approaches to linear programming

SIMPLEX algorithm
- classical approach (1940s)
- generalizes network simplex method for mincost flow
- generalizes Gaussian elimination
- exponential in worst case
- fast in practice

Ellipsoid methods
- new (1980s) algorithmic approach
- geometric divide-and-conquer
- polynomial time (!)

Interior-point methods
- make poly-time methods faster than simplex in practice
NP-completeness (quick review)

**P** is the class of decision problems that
- solvable on a deterministic machine in polynomial time

**NP** is the class of problems where solutions
- solvable on a nondeterministic machine in polynomial time

An **NP-hard problem** is
- one that every problem in NP easily reduces to

**NP-complete problems** are both in NP and NP-hard

**The main question:** Is \( P = NP \)?
- no Universal Problem-Solving Model exists unless \( P = NP \)

Current practice for NP-hard problems:
- assume that “trying all possibilities” is the best we can do
- hope that real problems are not worst-case instances
- work on easier versions
Perspective

LP is near the deep waters of NP-hardness
  - LP is in P (known for less than 20 years)
  - INTEGER programming is NP-complete

Cross-currents among fields of research
  - 1960s Operations research
  - 1970s Design and analysis of algorithms
  - 1980s Geometric algorithms

Specialized algorithms
  - may provide elegant optimal solution for all instances
    worth pursuing

General models (like mincost flow and LP)
  - may solve specific practical instances quickly
    worth pursuing

Universal Problem-Solving Model?