Linear Programming

- brewer's problem
- simplex algorithm
- implementation
- linear programming

References:
The Allocation of Resources by Linear Programming,
Scientific American, by Bob Bland
Algs in Java, Part 5

Overview: introduction to advanced topics

Main topics
- linear programming: the ultimate practical problem-solving model
- reduction: design algorithms, prove limits, classify problems
- NP: the ultimate theoretical problem-solving model
- combinatorial search: coping with intractability

Shifting gears
- from linear/quadratic to polynomial/exponential scale
- from individual problems to problem-solving models
- 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?
see ORF 307
- Quintessential tool for optimal allocation of scarce resources, among a number of competing activities.
- Powerful and general problem-solving method that encompasses:
  shortest path, network flow, MST, matching, assignment...
  Ax = b, 2-person zero sum games

Why significant?
- Widely applicable problem-solving model
- Dominates world of industry.
- Fast commercial solvers available: CPLEX, OSL.
- Powerful modeling languages available: AMPL, GAMS.
- Ranked among most important scientific advances of 20th century.

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.
Plasma physics. Optimal stellarator design.
Telecommunication. Network design, Internet routing.
Sports. Scheduling ACC basketball, handicapping horse races.

- **brewer's problem**
- simplex algorithm
- implementation
- linear programming

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.

	corn (lbs)	hops (oz)	malt (lbs)	profit ($)
available	480	160	1190	
ale (1 barrel)	5	4	35	13
beer (1 barrel)	15	4	20	23

Brewer's problem: choose product mix to maximize profits.

all ale (34 barrels)	179	136	1190	442
all beer (32 barrels)	480	128	640	736
20 barrels ale 20 barrels beer	400	160	1100	720
12 barrels ale 28 barrels beer	480	160	980	800
more profitable product mix?	?	?	?	>800 ?

34 barrels times 35 lbs malt per barrel is 1190 lbs [amount of available malt]

Brewer's problem: mathematical formulation

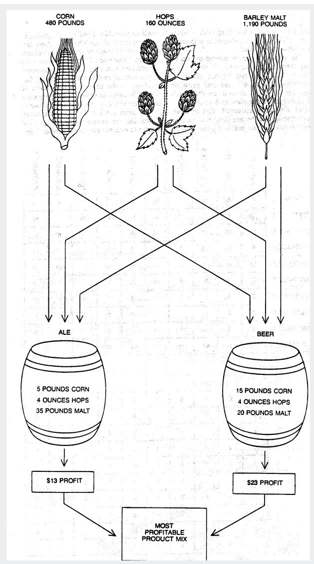
Small brewery produces ale and beer.

- Production limited by scarce resources: corn, hops, barley malt.
- Recipes for ale and beer require different proportions of resources.

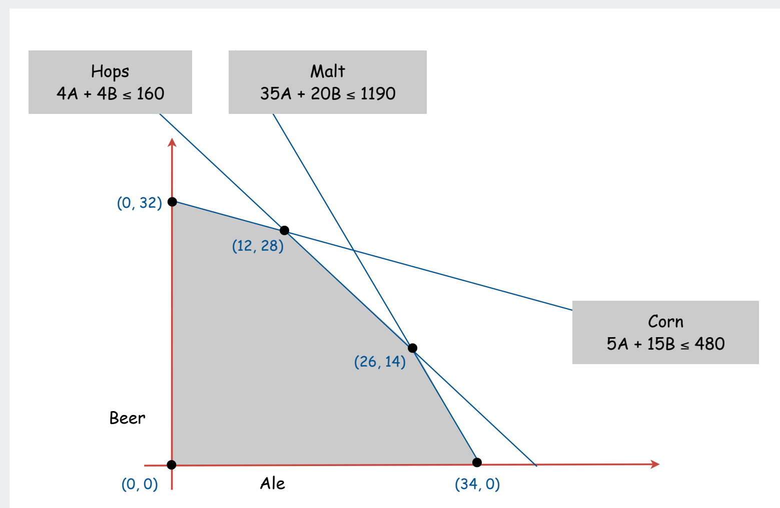
Mathematical formulation

- let A be the number of barrels of beer
- and B be the number of barrels of ale

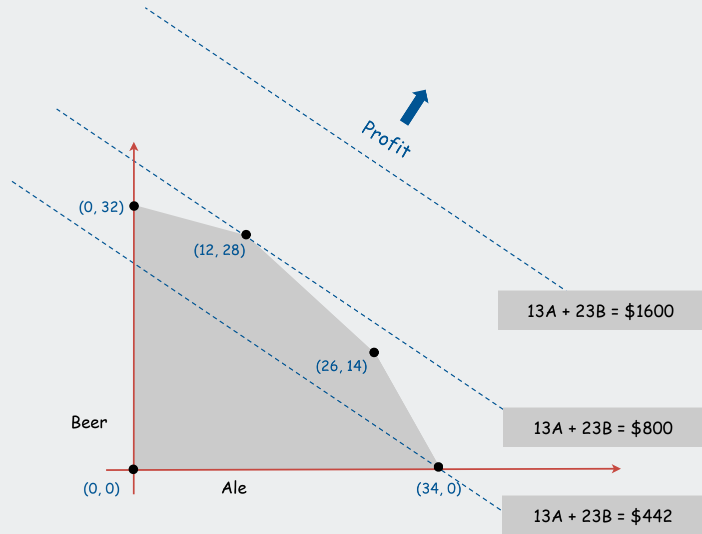
	ale		beer			
maximize	13A	+	23B			profit
subject to the constraints	5A	+	15B	≤	480	corn
	4A	+	4B	≤	160	hops
	35A	+	20B	≤	1190	malt
			A	≥	0	
			B	≥	0	



Brewer's problem: Feasible region

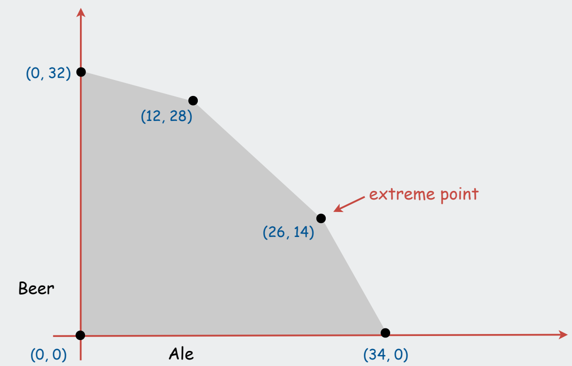


Brewer's problem: Objective function



Brewer's problem: Geometry

Brewer's problem observation. Regardless of objective function coefficients, an optimal solution occurs at an extreme point.



Standard form linear program

Input: real numbers a_{ij}, c_j, b_i.
Output: real numbers x_j.
n = # nonnegative variables, m = # constraints.
Maximize linear objective function subject to linear equations.

n variables

maximize	$c_1 x_1 + c_2 x_2 + \ldots + c_n x_n$
subject to the constraints (m equations)	$a_{11} x_1 + a_{12} x_2 + \ldots + a_{1n} x_n = b_1$
	$a_{21} x_1 + a_{22} x_2 + \ldots + a_{2n} x_n = b_2$
	...
	$a_{m1} x_1 + a_{m2} x_2 + \ldots + a_{mn} x_n = b_m$
	$x_1, x_2, \ldots, x_n \geq 0$

matrix version

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

"Linear" No x^2, xy, arccos(x), etc.
"Programming" " Planning" (term predates computer programming).

Converting the brewer's problem to the standard form

Original formulation

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

Standard form

- add variable Z and equation corresponding to objective function
- add slack variable to convert each inequality to an equality.
- now a 5-dimensional problem.

maximize	Z											
subject to the constraints	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								≥	0

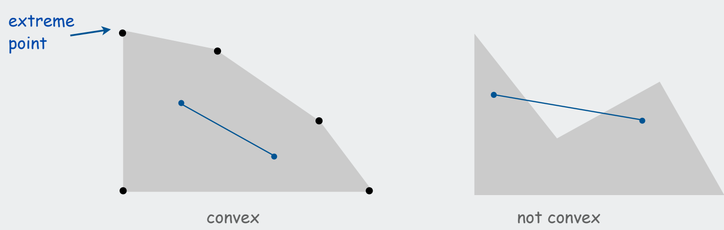
Geometry

A few principles from geometry:

- inequality: halfplane (2D), hyperplane (kD).
- bounded feasible region: convex polygon (2D), convex polytope (kD).

Convex set. If two points a and b are in the set, then so is $\frac{1}{2}(a + b)$.

Extreme point. 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.



Geometry (continued)

Extreme point property. If there exists an optimal solution to (P), then there exists one that is an extreme point.

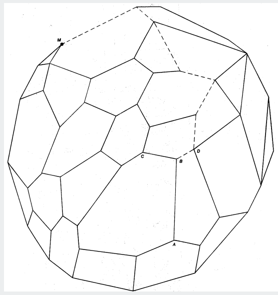
Good news. Only need to consider finitely many possible solutions.

Bad news. Number of extreme points can be exponential !

Ex: n-dimensional hypercube

Greedy property. Extreme point is optimal iff no neighboring extreme point is better.

local optima are global optima



- brewer's problem
- **simplex algorithm**
- implementation
- linear programming

Simplex Algorithm

Simplex algorithm. [George Dantzig, 1947]

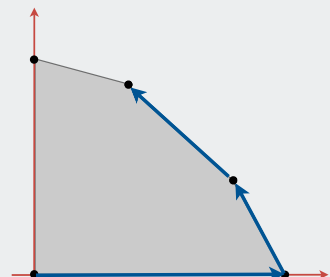
- Developed shortly after WWII in response to logistical problems, including Berlin airlift.
- One of greatest and most successful algorithms of all time.

Generic algorithm.

never decreasing objective function

- Start at some extreme point.
- Pivot from one extreme point to a neighboring one.
- Repeat until optimal.

How to implement? Linear algebra.

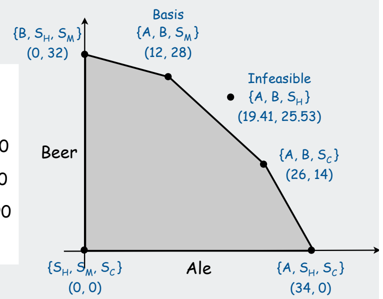


Simplex Algorithm: Basis

Basis. 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 solution $\Rightarrow$ BFS.
- BFS $\Leftrightarrow$ extreme point.

maximize	Z							
subject to the constraints	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					≥	0



Simplex Algorithm: Initialization

Start with slack variables as the basis.

Initial basic feasible solution (BFS).
- set non-basis variables A = 0, B = 0 (and Z = 0).
- 3 equations in 3 unknowns give S_C = 480, S_C = 160, S_C = 1190 (immediate).
- extreme point on simplex: origin

maximize	Z							
subject to the constraints	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					≥	0

basis = {S_C, S_H, S_M}
A = B = 0
Z = 0
S_C = 480
S_H = 160
S_M = 1190

Simplex Algorithm: Pivot 1

maximize	Z							
subject to the constraints	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					≥	0

basis = {S_C, S_H, S_M}
A = B = 0
Z = 0
S_C = 480
S_H = 160
S_M = 1190

Substitution B = (1/15)(480 – 5A – S_C) puts B into the basis ← which variable does it replace?

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

maximize	Z							
subject to the constraints	(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					≥	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

maximize	Z							
subject to the constraints	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					≥	0

basis = {S_C, S_H, S_M}
A = B = 0
Z = 0
S_C = 480
S_H = 160
S_M = 1190

Why pivot on B?
- Its objective function coefficient is positive
  (each unit increase in B from 0 increases objective value by $23)
- Pivoting on column 1 also OK.

Why pivot on row 2?
- Preserves feasibility by ensuring RHS ≥ 0.
- Minimum ratio rule: min { 480/15, 160/4, 1190/20 }.

Simplex Algorithm: Pivot 2

maximize Z

subject to the constraints

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

basis = $\{B, S_H, S_M\}$
$A = S_C = 0$
$Z = 736$
$B = 32$
$S_H = 32$
$S_M = 550$

Substitution $A = (3/8)(32 + (4/15)\,S_C - S_H)$ puts A into the basis
(rewrite 3nd equation, eliminate A in 1st, 2rd, and 4th equations)

maximize Z

subject to the constraints

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

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 all coefficients in top row are non-positive.

Q. Why is resulting solution optimal?
A. Any feasible solution satisfies system of equations in tableaux.
- 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.

maximize Z

subject to the constraints

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

basis = $\{A, B, S_M\}$
$S_C = S_H = 0$
$Z = 800$
$B = 28$
$A = 12$
$S_M = 110$

- brewer's problem
- simplex algorithm
- **implementation**
- linear programming

Simplex tableau

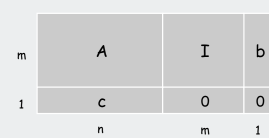
Encode standard form LP in a single Java 2D array

maximize Z

subject to the constraints

$$
\begin{array}{rcrcrcrcrcrcl}
13A & + & 23B & & & & & & & - & Z & = & 0 \\
5A & + & 15B & + & S_C & & & & & & & = & 480 \\
4A & + & 4B & & & + & S_H & & & & & = & 160 \\
35A & + & 20B & & & & & + & S_M & & & = & 1190 \\
 & & A,\ B,\ S_C,\ S_H,\ S_M & & & & & & & & & \geq & 0
\end{array}
$$

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 tableau

Encode standard form LP in a single Java 2D array (solution)

$$
\begin{array}{lrcl}
\text{maximize} & Z & & \\
\text{subject to the constraints} & -\ S_C - 2S_H - Z & = & -800 \\
& B + (1/10)\,S_C + (1/8)\,S_H & = & 28 \\
& A - (1/10)\,S_C + (3/8)\,S_H & = & 12 \\
& -\ (25/6)\,S_C - (85/8)\,S_H + S_M & = & 110 \\
& A,\ B,\ S_C, S_H, S_M & \geq & 0
\end{array}
$$

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

	n	m	1
m	A	I	b
1	c	0	0

Simplex algorithm transforms initial array into solution

Simplex algorithm: Bare-bones implementation

Construct the simplex tableau.

	n	m	1
m	A	I	b
1	c	0	0

```
public class Simplex
{
   private double[][] a;    // simplex tableaux
   private int M, N;

   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];
   }
```

constructor

put A[][] into tableau

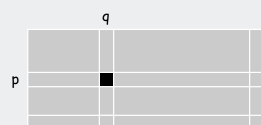
put I[] into tableau

put c[] into tableau

put b[] into tableau

Simplex algorithm: Bare-bones Implementation

Pivot on element (p, 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 && 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;
}
```

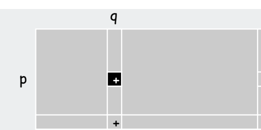
scale all elements but row p and column q

zero out column q

scale row p

Simplex Algorithm: Bare Bones Implementation

Simplex algorithm.




```
public void solve()
{
   while (true)
   {
      int p, q;
      for (q = 0; q < M + N; q++)
         if (a[M][q] > 0) break;
      if (q >= M + N) break;

      for (p = 0; p < M; p++)
         if (a[p][q] > 0) break;
      for (int i = p+1; i < M; i++)
         if (a[i][q] > 0)
            if (a[i][M+N] / a[i][q]
                  < a[p][M+N] / a[p][q])
               p = i;

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

find entering variable q (positive objective function coefficient)

find row p according to min ratio rule

min ratio test

Simplex Algorithm: Running Time

Remarkable property. In practice, simplex algorithm typically terminates after at most 2(m+n) pivots.

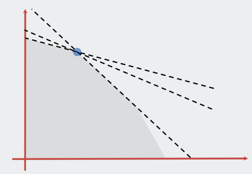
- No pivot rule that is guaranteed to be polynomial is known.
- Most pivot rules known to be exponential (or worse) in worst-case.

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

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 least index rule guarantees finite # of pivots.

Simplex Algorithm: Implementation Issues

To improve the bare-bones implementation

- Avoid stalling.
- Choose the pivot wisely.
- Watch for numerical stability.
- Maintain sparsity. ← requires fancy data structures
- Detect infeasiblity
- Detect unboundedness.
- Preprocess to reduce problem size.

Basic implementations available in many programming environments.

Commercial solvers routinely solve LPs with millions of variables.

LP solvers: basic implementations

Ex. 1: OR-Objects Java library

```
import drasys.or.mp.*;
import drasys.or.mp.lp.*;

public class LPDemo
{
   public static void main(String[] args) throws Exception
   {
      Problem prob = new Problem(3, 2);
      prob.getMetadata().put("lp.isMaximize", "true");
      prob.newVariable("x1").setObjectiveCoefficient(13.0);
      prob.newVariable("x2").setObjectiveCoefficient(23.0);
      prob.newConstraint("corn").setRightHandSide( 480.0);
      prob.newConstraint("hops").setRightHandSide( 160.0);
      prob.newConstraint("malt").setRightHandSide(1190.0);

      prob.setCoefficientAt("corn", "x1",  5.0);
      prob.setCoefficientAt("corn", "x2", 15.0);
      prob.setCoefficientAt("hops", "x1",  4.0);
      prob.setCoefficientAt("hops", "x2",  4.0);
      prob.setCoefficientAt("malt", "x1", 35.0);
      prob.setCoefficientAt("malt", "x2", 20.0);

      DenseSimplex lp = new DenseSimplex(prob);
      System.out.println(lp.solve());
      System.out.println(lp.getSolution());
   }
}
```

Ex. 2: MS Excel (!)

LP solvers: commercial strength

AMPL. [Fourer, Gay, Kernighan] An algebraic modeling language.
CPLEX solver. Industrial strength solver.

	ale		beer			
maximize	13A	+	23B			profit
subject to the constraints	5A	+	15B	≤	480	corn
	4A	+	4B	≤	160	hops
	35A	+	20B	≤	1190	malt
			A	≥	0	
			B	≥	0	

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];
```

separate data from model

```
[cos226:tucson] ~> ampl
AMPL Version 20010215 (SunOS 5.7)
ampl: model beer.mod;
ampl: data beer.dat;
ampl: solve;
CPLEX 7.1.0: optimal solution; objective 800
ampl: display x;
x [*] :=  ale 12   beer 28;
```

```
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;
```

beer.dat

History

1939. Production, planning. [Kantorovich]
1947. Simplex algorithm. [Dantzig]
1950. Applications in many fields.
1979. Ellipsoid algorithm. [Khachian]
1984. Projective scaling algorithm. [Karmarkar]
1990. Interior point methods.
- Interior point faster when polyhedron smooth like disco ball.
- Simplex faster when polyhedron spiky like quartz crystal.

200x. Approximation algorithms, large scale optimization.

- brewer's problem
- simplex algorithm
- implementation
- **linear programming**

Linear programming

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

easy part [omitted]:
convert to standard form

Examples:
- shortest paths
- maxflow
- bipartite matching
- .
- .
- .
- [a very long list]

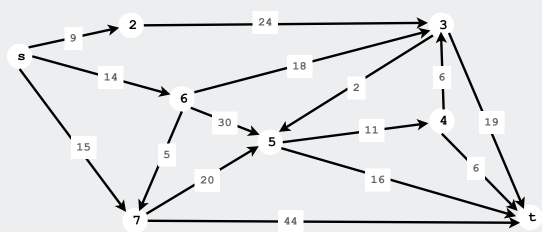
stay tuned [this lecture]

Single-source shortest-paths problem (revisited)

Given. Weighted digraph, single source s.

Distance from s to v: length of the shortest path from s to v .

Goal. Find distance (and shortest path) from s to every other vertex.



LP formulation of single-source shortest-paths problem

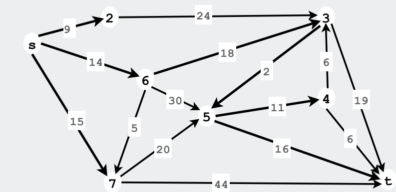
One variable per vertex, one inequality per edge.

minimize x_t

subject to the constraints

$$
\begin{aligned}
x_s + 9 &\le x_2\\
x_s + 14 &\le x_6\\
x_s + 15 &\le x_7\\
x_2 + 24 &\le x_3\\
x_3 + 2 &\le x_5\\
x_3 + 19 &\le x_t\\
x_4 + 6 &\le x_3\\
x_4 + 6 &\le x_t\\
x_5 + 11 &\le x_4\\
x_5 + 16 &\le x_t\\
x_6 + 18 &\le x_3\\
x_6 + 30 &\le x_5\\
x_6 + 5 &\le x_7\\
x_7 + 20 &\le x_5\\
x_7 + 44 &\le x_t\\
x_s &= 0\\
x_2, \ldots, x_t &\ge 0
\end{aligned}
$$

interpretation: x_i = length of shortest path from source to i



LP formulation of single-source shortest-paths problem

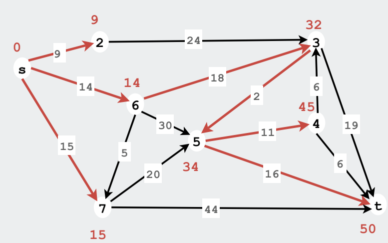
One variable per vertex, one inequality per edge.

minimize x_t

subject to the constraints

$$
\begin{aligned}
x_s + 9 &\le x_2\\
x_s + 14 &\le x_6\\
x_s + 15 &\le x_7\\
x_2 + 24 &\le x_3\\
x_3 + 2 &\le x_5\\
x_3 + 19 &\le x_t\\
x_4 + 6 &\le x_3\\
x_4 + 6 &\le x_t\\
x_5 + 11 &\le x_4\\
x_5 + 16 &\le x_t\\
x_6 + 18 &\le x_3\\
x_6 + 30 &\le x_5\\
x_6 + 5 &\le x_7\\
x_7 + 20 &\le x_5\\
x_7 + 44 &\le x_t\\
x_s &= 0\\
x_2, \ldots, x_t &\ge 0
\end{aligned}
$$

interpretation: x_i = length of shortest path from source to i

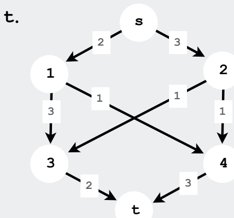


solution

$$
\begin{aligned}
x_s &= 0\\
x_2 &= 9\\
x_3 &= 32\\
x_4 &= 45\\
x_5 &= 34\\
x_6 &= 14\\
x_7 &= 15\\
x_t &= 50
\end{aligned}
$$

Maxflow problem

Given: Weighted digraph, source s, destination t.

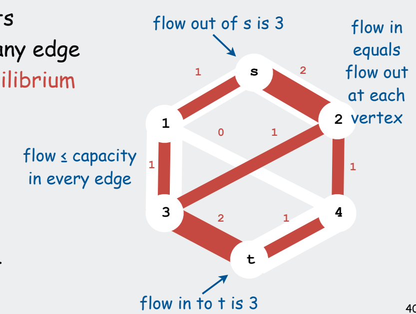


Interpret edge weights as capacities

- Models material flowing through network
- Ex: oil flowing through pipes
- Ex: goods in trucks on roads
- [many other examples]

Flow: A different set of edge weights

- flow does not exceed capacity in any edge
- flow at every vertex satisfies equilibrium
  [flow in equals flow out]



Goal: Find maximum flow from s to t

LP formulation of maxflow problem

One variable per edge.
One inequality per edge, one equality per vertex.

maximize x_{ts}

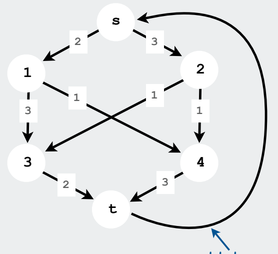
subject to the constraints

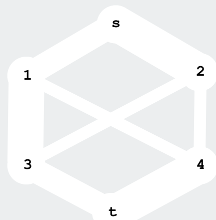
$$\left.\begin{aligned} x_{s1} &\le 2 \\ x_{s2} &\le 3 \\ x_{13} &\le 3 \\ x_{14} &\le 1 \\ x_{23} &\le 1 \\ x_{24} &\le 1 \\ x_{3t} &\le 2 \\ x_{4t} &\le 3 \end{aligned}\right\}\ \text{capacity constraints}$$

$$\text{equilibrium constraints}\ \left\{\begin{aligned} x_{ts} &= x_{s1} + x_{s2} \\ x_{s1} &= x_{13} + x_{14} \\ x_{s2} &= x_{23} + x_{24} \\ x_{13} + x_{23} &= x_{3t} \\ x_{14} + x_{24} &= x_{4t} \\ x_{3t} + x_{4t} &= x_{ts} \end{aligned}\right.$$

all $x_{ij} \ge 0$

interpretation:
x_{ij} = flow in edge i-j





LP formulation of maxflow problem

One variable per edge.
One inequality per edge, one equality per vertex.

maximize x_{ts}

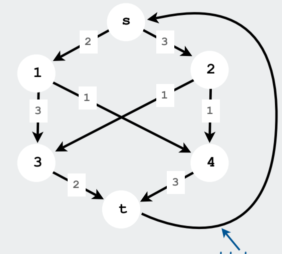
subject to the constraints

$$\left.\begin{aligned} x_{s1} &\le 2 \\ x_{s2} &\le 3 \\ x_{13} &\le 3 \\ x_{14} &\le 1 \\ x_{23} &\le 1 \\ x_{24} &\le 1 \\ x_{3t} &\le 2 \\ x_{4t} &\le 3 \end{aligned}\right\}\ \text{capacity constraints}$$

$$\text{equilibrium constraints}\ \left\{\begin{aligned} x_{ts} &= x_{s1} + x_{s2} \\ x_{s1} &= x_{13} + x_{14} \\ x_{s2} &= x_{23} + x_{24} \\ x_{13} + x_{23} &= x_{3t} \\ x_{14} + x_{24} &= x_{4t} \\ x_{3t} + x_{4t} &= x_{ts} \end{aligned}\right.$$

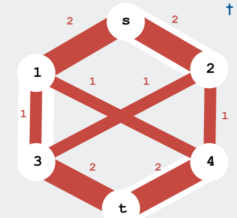
all $x_{ij} \ge 0$

interpretation:
x_{ij} = flow in edge i-j



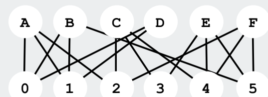
solution

$x_{s1} = 2$
$x_{s2} = 2$
$x_{13} = 1$
$x_{14} = 1$
$x_{23} = 1$
$x_{24} = 1$
$x_{3t} = 2$
$x_{4t} = 2$
$x_{ts} = 4$ ← maxflow value



Maximum cardinality bipartite matching problem

Given: Two sets of vertices, set of edges
(each connecting one vertex in each set)



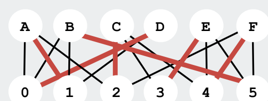
Matching: set of edges
with no vertex appearing twice

Interpretation: mutual preference constraints
- Ex: people to jobs
- Ex: medical students to residence positions
- Ex: students to writing seminars
- [many other examples]

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

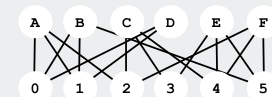
Example: Job offers

Goal: find a maximum cardinality matching



LP formulation of maximum cardinality bipartite matching problem

One variable per edge, one equality per vertex.



maximize $x_{A0} + x_{A1} + x_{A2} + x_{B0} + x_{B1} + x_{B5} + x_{C2} + x_{C3} + x_{C4} + x_{D0} + x_{D1} + x_{E3} + x_{E4} + x_{E5} + x_{F2} + x_{F4} + x_{F5}$

subject to the constraints

$$\left.\begin{aligned} x_{A0} + x_{A1} + x_{A2} &= 1 \\ x_{B0} + x_{B1} + x_{B5} &= 1 \\ x_{C2} + x_{C3} + x_{C4} &= 1 \\ x_{D0} + x_{D1} &= 1 \\ x_{E3} + x_{E4} + x_{E5} &= 1 \\ x_{F2} + x_{F4} + x_{F5} &= 1 \end{aligned}\right\}\ \text{constraints on top vertices}$$

$$\left.\begin{aligned} x_{A0} + x_{B0} + x_{D0} &= 1 \\ x_{A1} + x_{B1} + x_{D1} &= 1 \\ x_{A2} + x_{C2} + x_{F2} &= 1 \\ x_{C3} + x_{E3} &= 1 \\ x_{C4} + x_{E4} + x_{F4} &= 1 \\ x_{B5} + x_{E5} + x_{F5} &= 1 \end{aligned}\right\}\ \text{constraints on bottom vertices}$$

all $x_{ij} \ge 0$

interpretation:
An edge is in the matching iff $x_{ij} = 1$

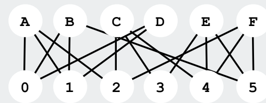
Crucial point:
not always so lucky!

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

Corollary. Can solve bipartite matching problem by solving LP

LP formulation of maximum cardinality bipartite matching problem

One variable per edge, one equality per vertex.



maximize
$$x_{A0} + x_{A1} + x_{A2} + x_{B0} + x_{B1} + x_{B5} + x_{C2} + x_{C3} + x_{C4} + x_{D0} + x_{D1} + x_{E3} + x_{E4} + x_{E5} + x_{F2} + x_{F4} + x_{F5}$$

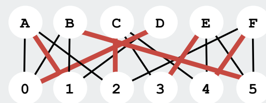
subject to the constraints

$$x_{A0} + x_{A1} + x_{A2} = 1$$
$$x_{B0} + x_{B1} + x_{B5} = 1$$
$$x_{C2} + x_{C3} + x_{C4} = 1$$
$$x_{D0} + x_{D1} = 1$$
$$x_{E3} + x_{E4} + x_{E5} = 1$$
$$x_{F2} + x_{F4} + x_{F5} = 1$$
$$x_{A0} + x_{B0} + x_{D0} = 1$$
$$x_{A1} + x_{B1} + x_{D1} = 1$$
$$x_{A2} + x_{C2} + x_{F2} = 1$$
$$x_{C3} + x_{E3} = 1$$
$$x_{C4} + x_{E4} + x_{F4} = 1$$
$$x_{B5} + x_{E5} + x_{F5} = 1$$
$$\text{all } x_{ij} \geq 0$$

interpretation: An edge is in the matching iff $x_{ij} = 1$

solution

$x_{A1} = 1$
$x_{B5} = 1$
$x_{C2} = 1$
$x_{D0} = 1$
$x_{E3} = 1$
$x_{F4} = 1$
all other $x_{ij} = 0$



Linear programming perspective

Got an optimization problem?
ex: shortest paths, maxflow, matching, . . . [many, many, more]

Approach 1: Use a specialized algorithm to solve it
- Algs in Java
- vast literature on complexity
- performance on real problems not always well-understood

Approach 2: Use linear programming
- a direct mathematical representation of the problem often works
- immediate solution to the problem at hand is often available
- might miss specialized solution, but might not care

Got an LP solver? Learn to use it!

```
[cos226:tucson] ~> ampl
AMPL Version 20010215 (SunOS 5.7)
ampl: model maxflow.mod;
ampl: data maxflow.dat;
ampl: solve;
CPLEX 7.1.0: optimal solution;
objective 4;
```


LP: the ultimate problem-solving model (in practice)

Fact 1: Many practical problems are easily formulated as LPs
Fact 2: Commercial solvers can solve those LPs quickly

More constraints on the problem?
- specialized algorithm may be hard to fix
- can just add more inequalities to LP

← Ex. Mincost maxflow and other generalized versions

New problem?
- may not be difficult to formulate LP
- may be very difficult to develop specialized algorithm

Today's problem?
- similar to yesterday's
- edit tableau, run solver

← Ex. Airline scheduling [similar to vast number of other business processes]

Too slow?
- could happen
- doesn't happen

Want to learn more? ORFE 307

Ultimate problem-solving model (in theory)

Is there an ultimate problem-solving model?
- Shortest paths
- Maximum flow
- Bipartite matching
- . . .
- Linear programming

} tractable

- .
- .
- .
- NP-complete problems
- .
- .
- .

} intractable ?

[see next lecture]

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

Want to learn more? COS 423

LP perspective

LP is near the deep waters of intractability.

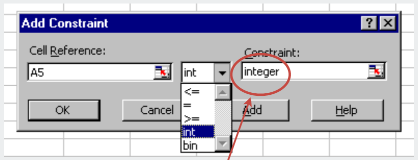
Good news:

- LP has been widely used for large practical problems for 50+ years
- Existence of guaranteed poly-time algorithm known for 25+ years.

Bad news:

constrain variables to have integer values

- Integer linear programming is NP-complete
- (existence of guaranteed poly-time algorithm is highly unlikely).
- [stay tuned]



An unsuspecting MBA student transitions to
the world of intractability with a single mouse click.