6. Linear Programming

- brewer's problem
- simplex algorithm
- implementations
- duality
- modeling

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, ...

to learn much much more, see ORF 307

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

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.

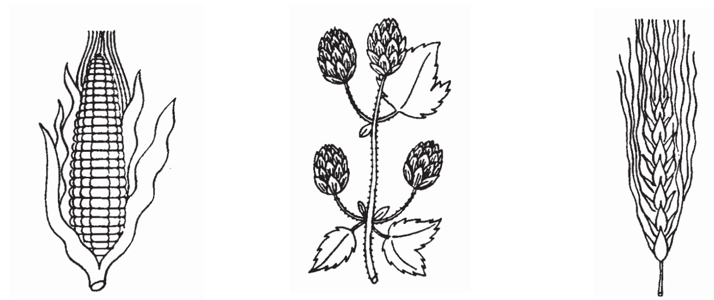
- **brewer's problem**
- simplex algorithm
- implementations
- duality
- modeling

The Allocation of Resources by Linear Programming by Robert Bland,
Scientific American, Vol. 244, No. 6, June 1981.

Toy LP example: brewer's problem

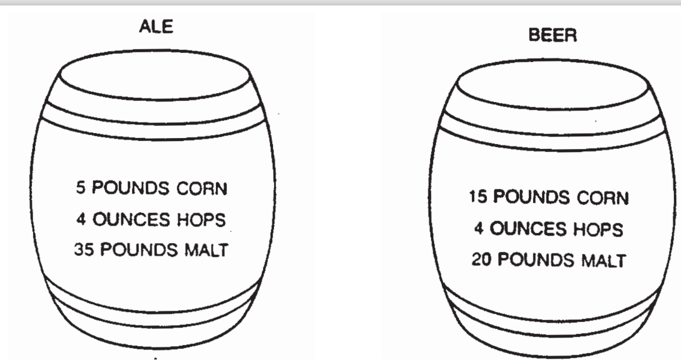
Small brewery produces ale and beer.

- Production limited by scarce resources: corn, hops, barley malt.



corn (480 lbs) hops (160 oz) malt (1190 lbs)

- Recipes for ale and beer require different proportions of resources.



\$13 profit per barrel \$23 profit per barrel

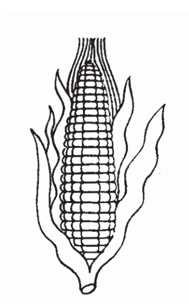
Toy LP example: brewer's problem

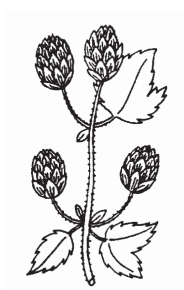
Brewer's problem: choose product mix to maximize profits.

34 barrels × 35 lbs malt = 1190 lbs
[amount of available malt]

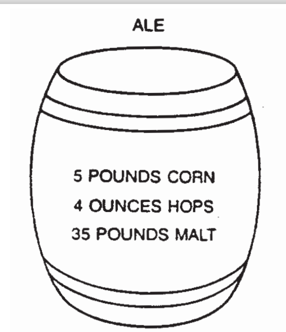
ale	beer	corn	hops	malt	profit
34	0	179	136	1190	$442
0	32	480	128	640	$736
19.5	20.5	405	160	1092.5	$725
12	28	480	160	980	$800
?	?				> $800 ?

good are indivisible

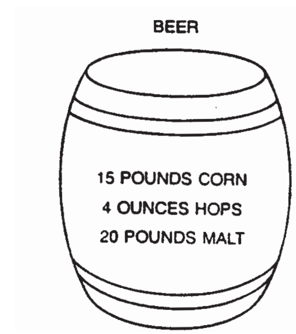

corn (480 lbs)


hops (160 oz)


malt (1190 lbs)



$13 profit per barrel



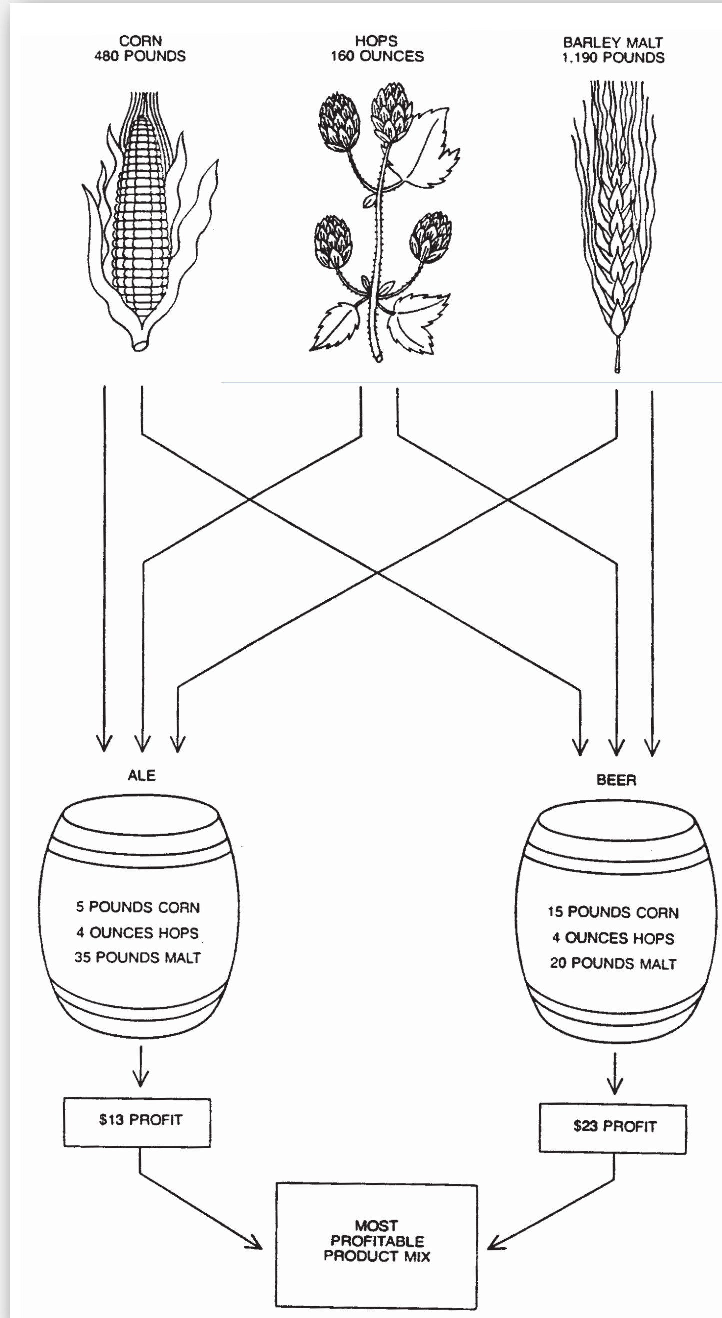
$23 profit per barrel

Brewer's problem: linear programming formulation

Linear programming formulation.

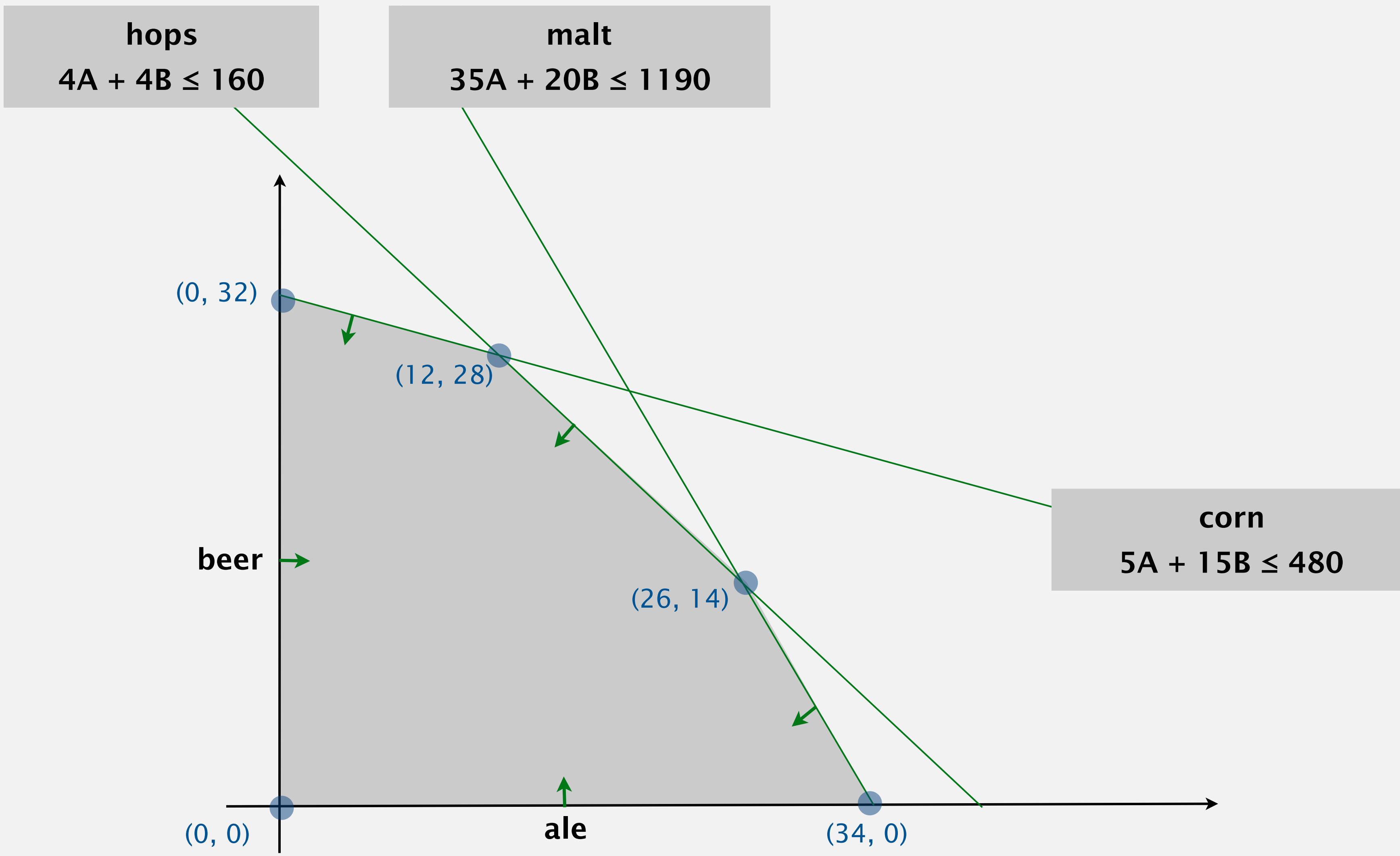
- Let A be the number of barrels of ale.
- Let B be the number of barrels of beer.

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

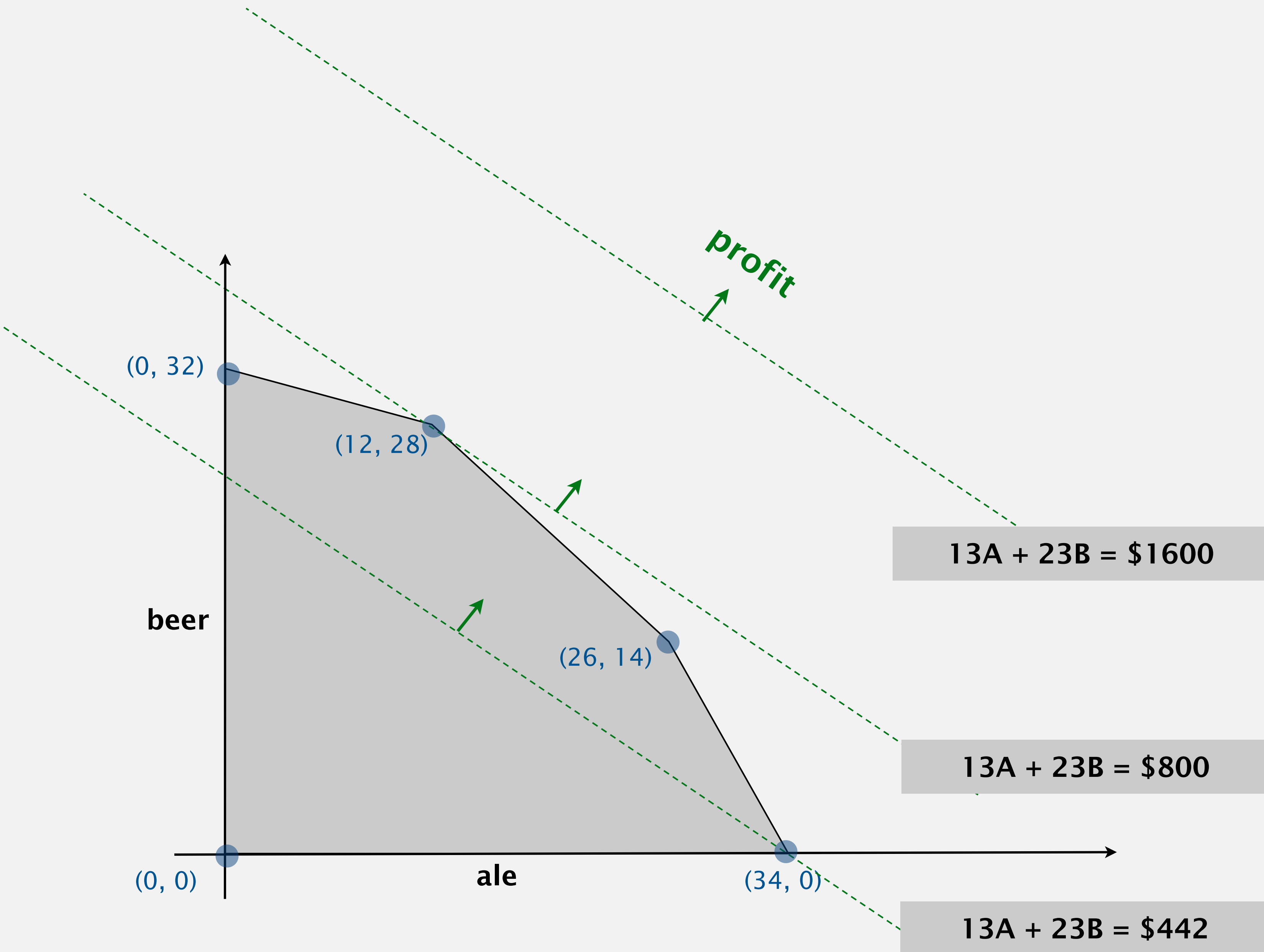


Brewer's problem: feasible region

Inequalities define halfplanes; feasible region is a convex polygon.



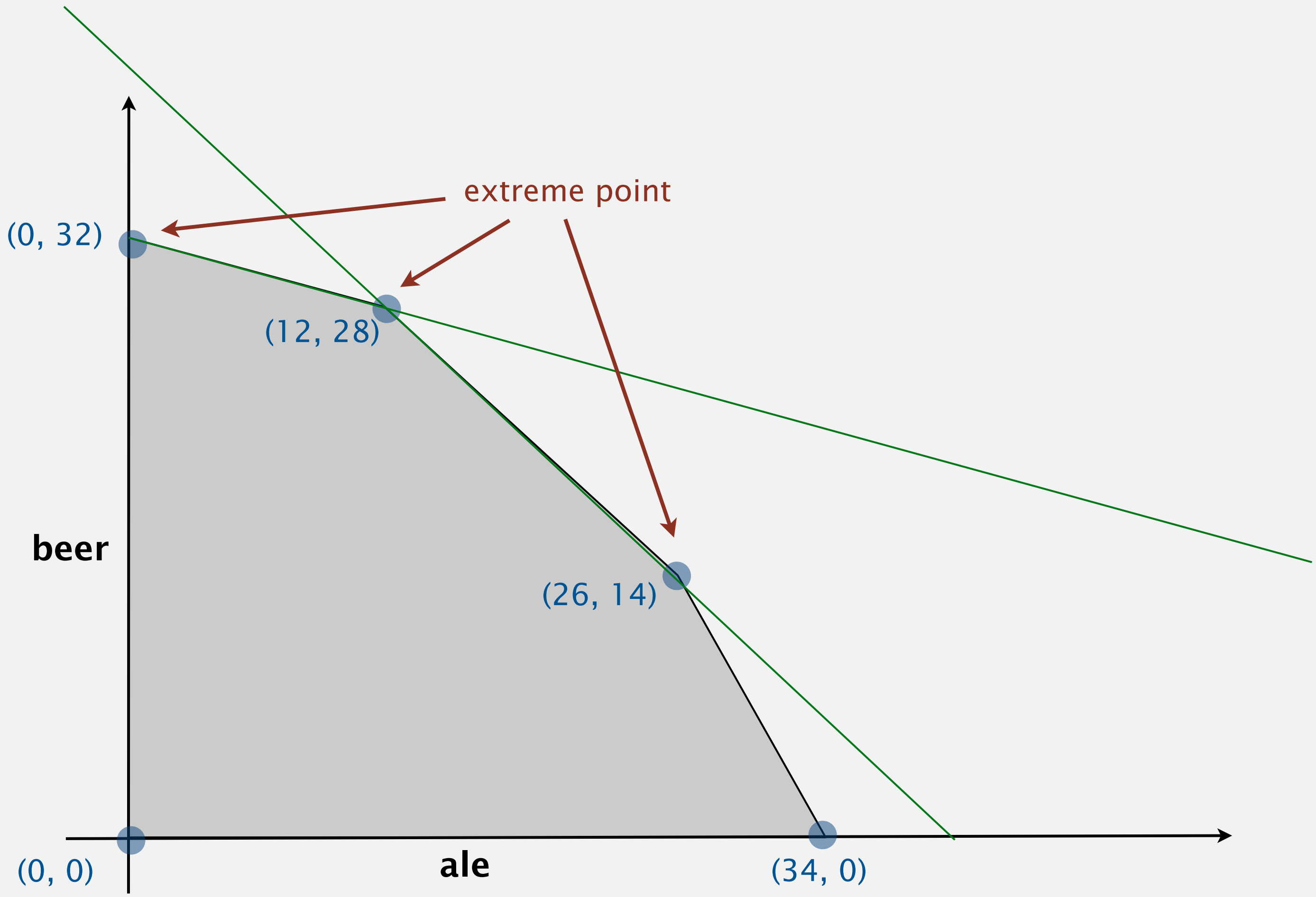
Brewer's problem: objective function



Brewer's problem: geometry

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

intersection of 2 constraints in 2d



Standard form linear program

Goal. Maximize linear objective function of n nonnegative variables, subject to m linear equations.

linear means no x^2, xy, $\arccos(x)$, etc.

- Input: real numbers a_{ij}, c_j, b_i.
- Output: real numbers x_j.

primal problem (P)

$$
\begin{array}{llcl}
\text{maximize} & c_1 x_1 + c_2 x_2 + \dots + c_n x_n & & \\
\text{subject to the constraints} & a_{11} x_1 + a_{12} x_2 + \dots + a_{1n} x_n & = & b_1 \\
 & a_{21} x_1 + a_{22} x_2 + \dots + a_{2n} x_n & = & b_2 \\
 & \vdots & & \vdots \\
 & a_{m1} x_1 + a_{m2} x_2 + \dots + a_{mn} x_n & = & b_m \\
 & x_1 \,,\; x_2 \,,\; \dots \,,\; x_n & \ge & 0
\end{array}
$$

matrix version

$$
\begin{array}{ll}
\text{maximize} & c^T x \\
\text{subject to the constraints} & A x = b \\
 & x \ge 0
\end{array}
$$

Converting the brewer's problem to the standard form

Original formulation.

$$
\begin{array}{lrcrcr}
\text{maximize} & 13A & + & 23B & & \\
\text{subject to the constraints} & 5A & + & 15B & \le & 480 \\
 & 4A & + & 4B & \le & 160 \\
 & 35A & + & 20B & \le & 1190 \\
 & A & , & B & \ge & 0
\end{array}
$$

Standard form.

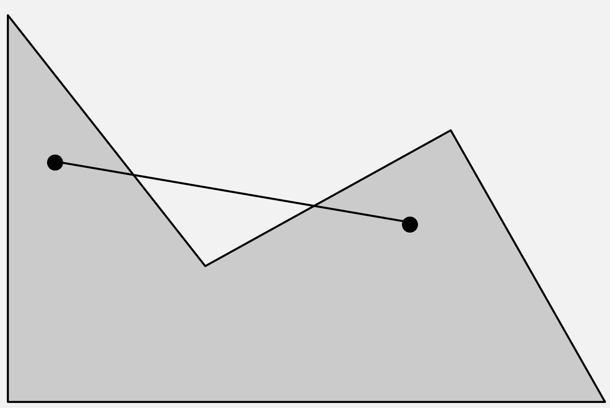
- Add variable Z and equation corresponding to objective function.
- Add slack variable to convert each inequality to an equality.
- Now a 6-dimensional problem.

$$
\begin{array}{lrcrcccccccccr}
\text{maximize} & Z & & & & & & & & & & & & \\
 & 13A & + & 23B & & & & & & & - & Z & = & 0 \\
\hline
\text{subject to the constraints} & 5A & + & 15B & + & S_C & & & & & & & = & 480 \\
 & 4A & + & 4B & & & + & S_H & & & & & = & 160 \\
 & 35A & + & 20B & & & & & + & S_M & & & = & 1190 \\
 & A & , & B & , & S_C & , & S_C & , & S_M & & & \ge & 0
\end{array}
$$

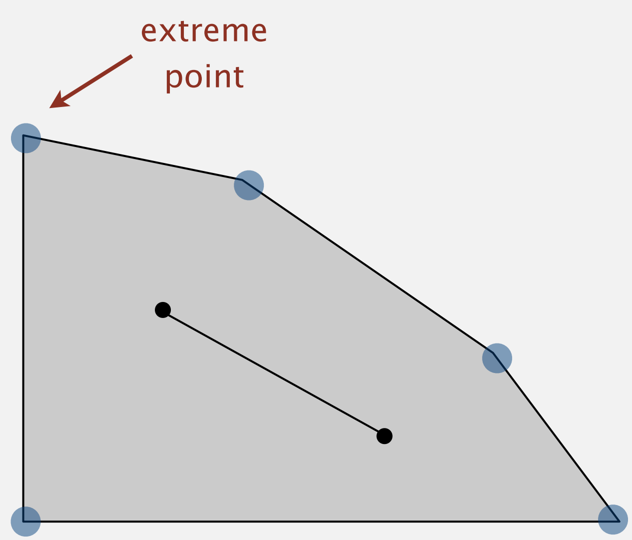
Inequalities define halfspaces; feasible region is a convex polyhedron.

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.



not convex



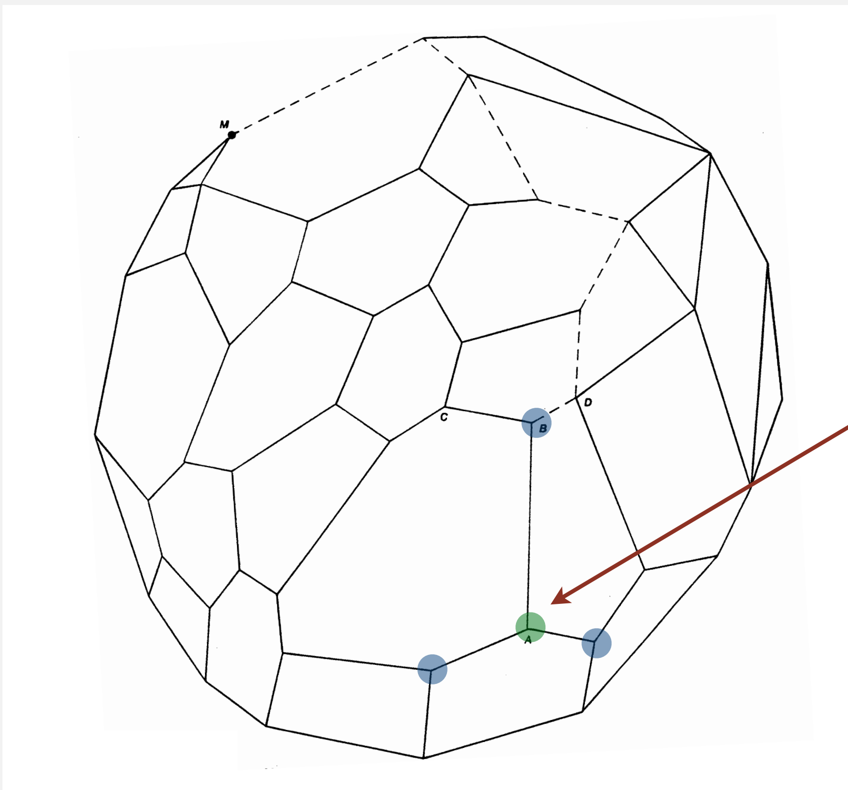
convex

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!



local optima are global optima
(follows because objective function is linear
and feasible region is convex)

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

- brewer's problem
- **simplex algorithm**
- implementations
- duality
- modeling

Simplex algorithm. [George Dantzig, 1947]

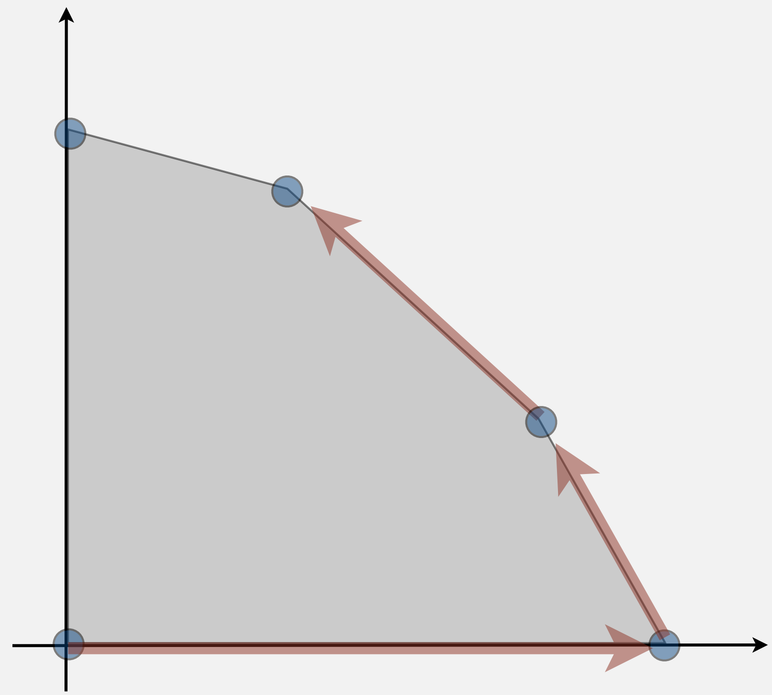
- Developed shortly after WWII in response to logistical problems, including Berlin airlift.
- Ranked as one of top 10 scientific algorithms of 20th century.

Generic algorithm.

- Start at some extreme point.
- Pivot from one extreme point to an adjacent one.
- Repeat until optimal.

never decreasing objective function

How to implement? Linear algebra.



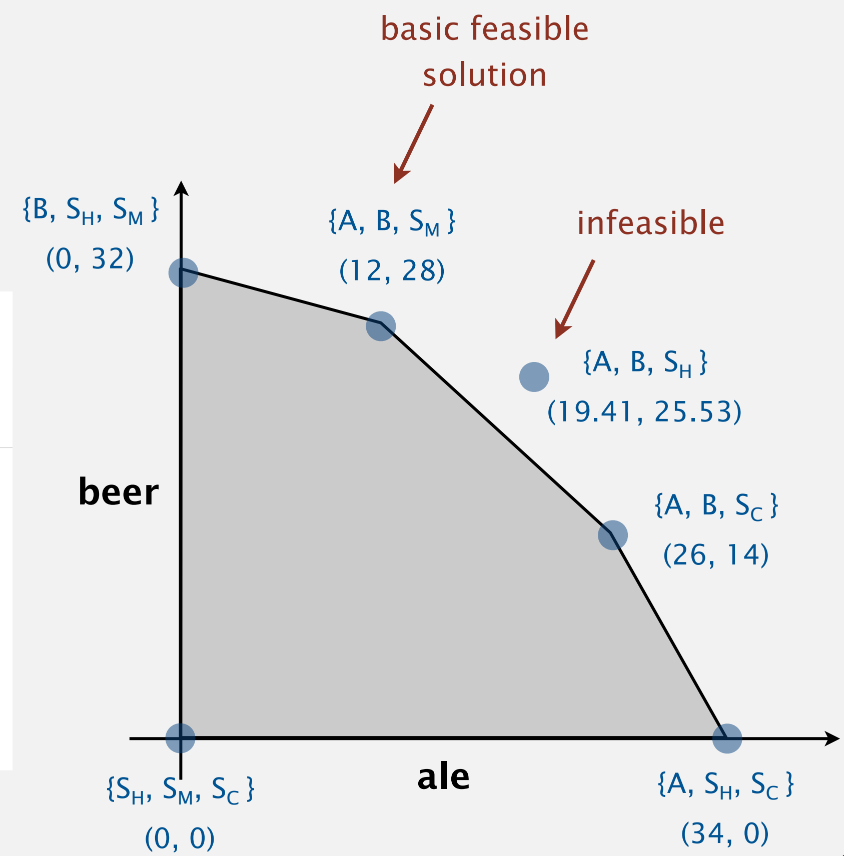
Simplex algorithm: basis

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

maximize	Z
	$13A + 23B - Z = 0$
subject to the constraints	$5A + 15B + S_C = 480$
	$4A + 4B + S_H = 160$
	$35A + 20B + S_M = 1190$
	$A, B, S_C, S_H, S_M \geq 0$



Simplex algorithm: initialization

maximize	Z															
	13A	+	23B									–	Z	=	0
subject to the constraints	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

Initial basic feasible solution.

one basic variable per row

- 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 $S_C = 480$, $S_H = 160$, $S_M = 1190$.

no algebra needed

Simplex algorithm: pivot 1

maximize Z

subject to the constraints

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

pivot

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

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)

which basic variable does B replace?

maximize Z

subject to the constraints

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

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

Simplex algorithm: pivot 1

positive coefficient

pivot

$$
\begin{array}{llllllllllll}
\text{maximize} & Z & & & & & & & & & & \\
 & 13A & + & 23B & & & & & - & Z & = & 0 \\
\hline
\text{subject} & 5A & + & 15B & + & S_C & & & & & = & 480 \\
\text{to the} & 4A & + & 4B & & & + & S_H & & & = & 160 \\
\text{constraints} & 35A & + & 20B & & & & & + & S_M & = & 1190 \\
 & A & , & B & , & S_C & , & S_H & , & S_M & \geq & 0
\end{array}
$$

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 $\text{RHS} \geq 0$.
- Minimum ratio rule: $\min \{ 480/15,\ 160/4,\ 1190/20 \}$.

Simplex algorithm: pivot 2

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

pivot

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

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)

which basic variable does A replace?

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

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 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.

maximize	Z												
				$-$	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 }

$S_C = S_H = 0$

$Z = 800$

$B = 28$

$A = 12$

$S_M = 110$

- brewer's problem
- simplex algorithm
- implementations
- duality
- modeling

Simplex tableau

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

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

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

$$
\begin{array}{c|c|c|c}
m & A & I & b \\
1 & c & 0 & 0 \\
 & n & m & 1
\end{array}
$$

initial simplex tableaux

Simplex tableau

Simplex algorithm transforms initial 2D array into solution.

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

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

final simplex tableaux

$$
\begin{array}{c|c|c|c|}
m & & & \\
\hline
1 & \leq 0 & \leq 0 & -Z^* \\
\hline
\multicolumn{1}{c}{} & \multicolumn{1}{c}{n} & \multicolumn{1}{c}{m} & \multicolumn{1}{c}{1}
\end{array}
$$

Simplex algorithm: initial simplex tableaux

Construct the initial simplex tableau.

m	A	I	b
1	c	0	0
	n	m	1

```
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];
   }
```

constructor

put `A[][]` into tableau

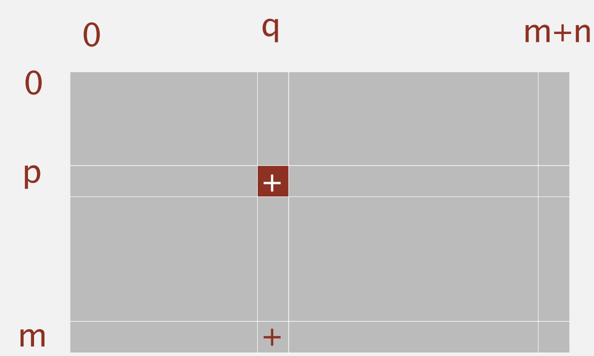
put `I[][]` into tableau

put `c[]` into tableau

put `b[]` into tableau

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;
}
```

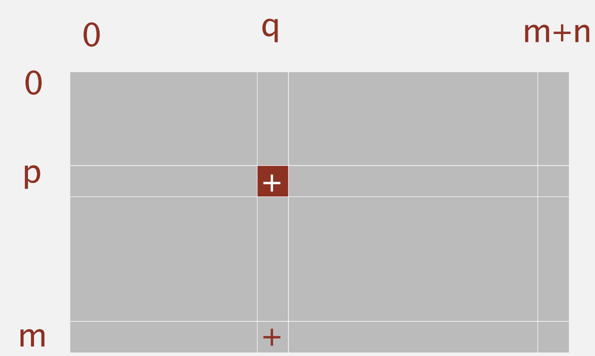
entering column q has positive objective function coefficient

optimal

Simplex algorithm: min-ratio rule

Find leaving row p using min ratio rule.

(Bland's rule: if a tie, choose first such row)




```
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;
}
```

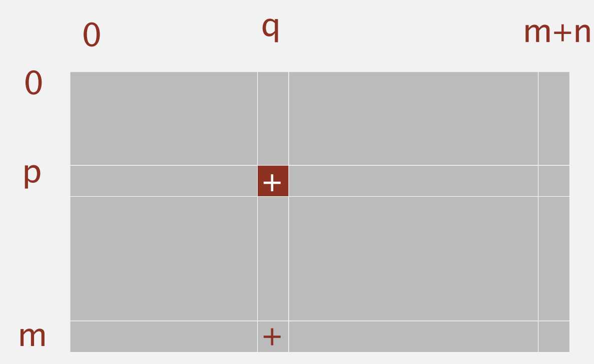
leaving row

consider only positive entries

row p has min ratio so far

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 && 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;
}
```

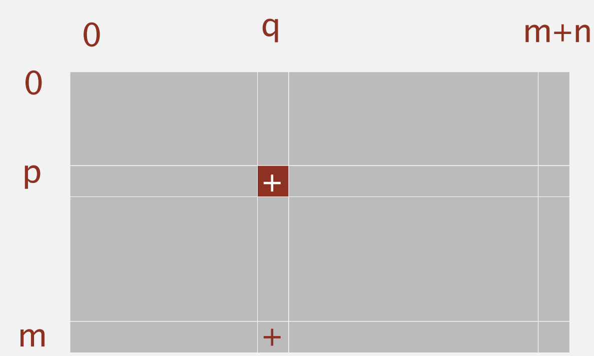
scale all entries but row p and column q

zero out column q

scale row p

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);
   }
}
```

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.

- 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

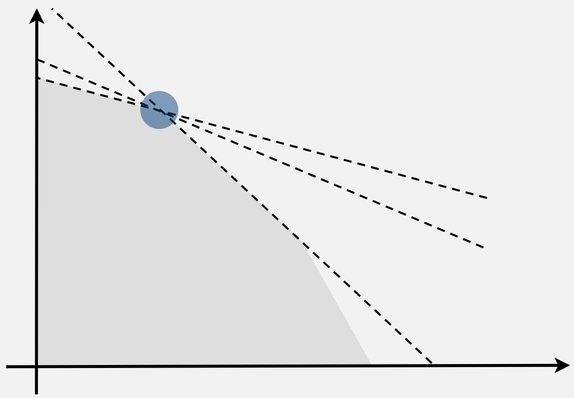
Daniel A. Spielman*
Department of Mathematics
M.I.T.
Cambridge, MA 02139
spielman@mit.edu

Shang-Hua Teng†
Akamai Technologies Inc. and
Department of Computer Science
University of Illinois at Urbana-Champaign
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.

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

http://or-objects.org/app/library

```
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.getMetadata().put("lp.isMaximize", "true");
      problem.newVariable("x1").setObjectiveCoefficient(13.0);
      problem.newVariable("x2").setObjectiveCoefficient(23.0);
      problem.newConstraint("corn").setRightHandSide( 480.0);
      problem.newConstraint("hops").setRightHandSide( 160.0);
      problem.newConstraint("malt").setRightHandSide(1190.0);

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

`http://www2.isye.gatech.edu/~wcook/qsopt`

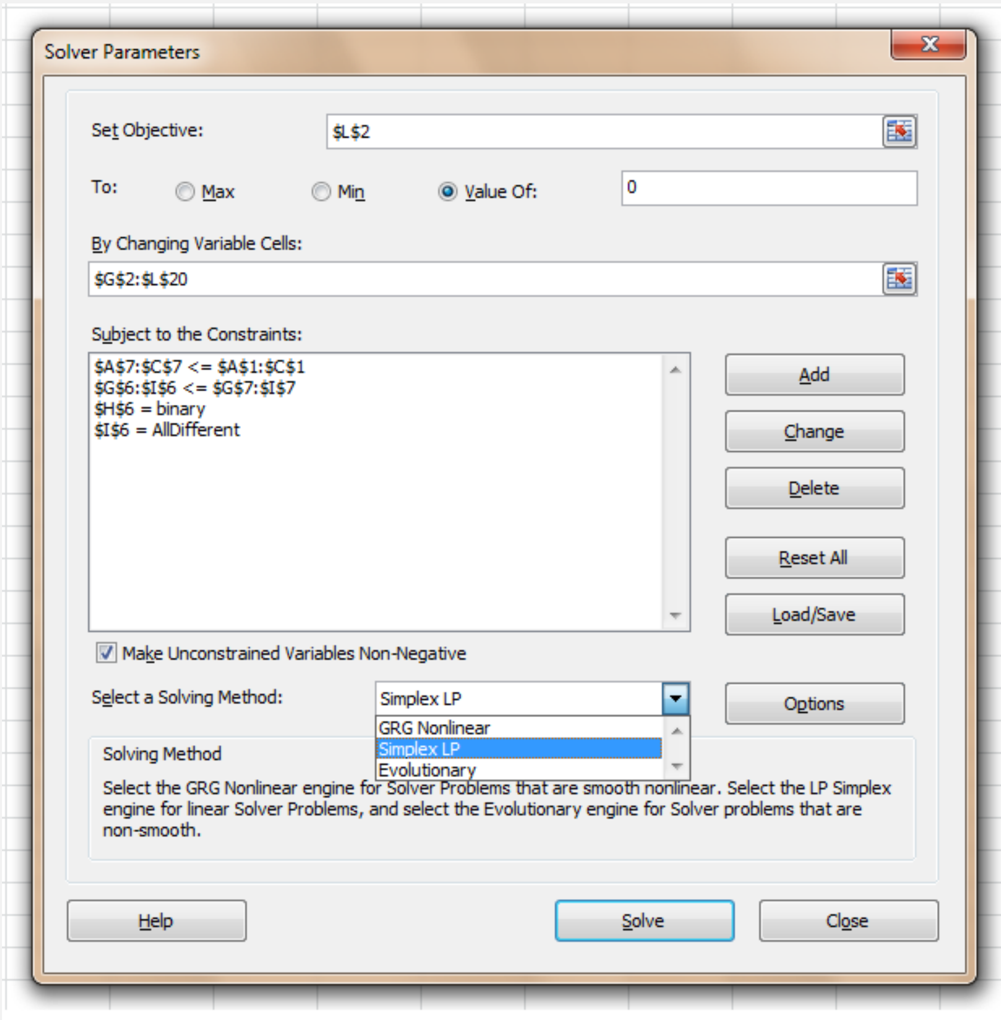
```
import qs.*;

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);
   }
}
```

```
% more beer.lp
Problem
   Beer
Maximize
  profit: 13A + 23B
Subject
    corn:  5A + 15B <=  480.0
    hops:  4A +  4B <=  160.0
    malt: 35A + 20B <= 1190.0
End
```

problem in LP or MPS format

```
% java -cp .:qsopt.jar QSoptSolver beer.lp
Optimal profit = 800.0
Optimal primal solution:
  A = 12.000000
  B = 28.000000
```

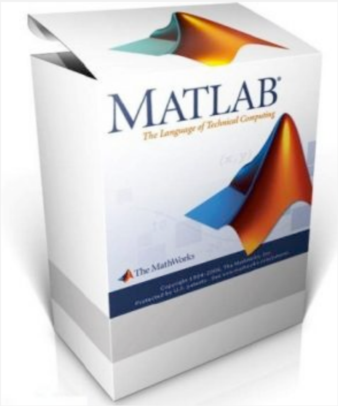
Ex 3. Microsoft Excel Solver add-in solves linear programs.



sorry, no longer support on Mac

Ex 4. Matlab command `linprog` in optimization toolbox solves LPs.

```
>> 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)
x =
    12.0000
    28.0000
```



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.

but license costs $$$

```
[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;
```

```
% more 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];
```

```
% more beer.dat
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;
```

- brewer's problem
- simplex algorithm
- implementations
- duality
- modeling

LP duality: economic interpretation

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

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

$A^* = 12$
$B^* = 28$
$OPT = 800$

coincidence?

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

$C^* = 1$
$H^* = 2$
$M^* = 0$
$OPT = 800$

LP duality: sensitivity analysis

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!

maximize Z

subject to the constraints

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

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)

$$
\begin{array}{llllllllll}
\max & c_1 x_1 & + & c_2 x_2 & + \ldots + & c_n x_n & & \\
\text{subject to} & a_{11} x_1 & + & a_{12} x_2 & + \ldots + & a_{1n} x_n & = & b_1 \\
& a_{21} x_1 & + & a_{22} x_2 & + \ldots + & a_{2n} x_n & = & b_2 \\
& \vdots & & \vdots & \vdots & \vdots & & \vdots \\
& a_{m1} x_1 & + & a_{m2} x_2 & + \ldots + & a_{mn} x_n & = & b_m \\
& x_1 & , & x_2 & , \ldots , & x_n & \geq & 0
\end{array}
$$

dual problem (D)

$$
\begin{array}{llllllllll}
\min & b_1 y_1 & + & b_2 y_2 & + \ldots + & b_m y_m & & \\
\text{subject to} & a_{11} y_1 & + & a_{21} y_2 & + \ldots + & a_{n1} y_m & = & c_1 \\
& a_{12} y_1 & + & a_{22} y_2 & + \ldots + & a_{n2} y_m & = & c_2 \\
& \vdots & & \vdots & \vdots & \vdots & & \vdots \\
& a_{1n} y_1 & + & a_{2n} y_2 & + \ldots + & a_{nm} y_m & = & c_n \\
& y_1 & , & y_2 & , \ldots , & y_m & \geq & 0
\end{array}
$$

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 the constraints	$A x = b$
	$x \geq 0$

dual problem (D)

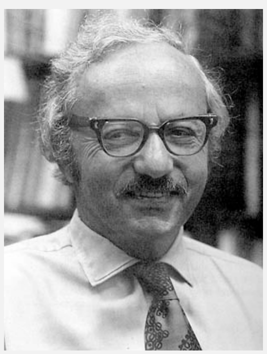
minimize	$b^T y$
subject to the constraints	$A^T y \geq c$
	$y \geq 0$

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

Brief history

1939. Production, planning. [Kantorovich]
1947. Simplex algorithm. [Dantzig]
1947. Duality. [von Neumann, Dantzig, Gale-Kuhn-Tucker]
1947. Equilibrium theory. [Koopmans]
1948. Berlin airlift. [Dantzig]
1975. Nobel Prize in Economics. [Kantorovich and Koopmans]
1979. Ellipsoid algorithm. [Khachiyan]
1984. Projective-scaling algorithm. [Karmarkar]
1990. Interior-point methods. [Nesterov-Nemirovskii, Mehorta, ...]


Kantorovich


George Dantzig


von Neumann


Koopmans


Khachiyan


Karmarkar

- brewer's problem
- simplex algorithm
- implementations
- duality
- **modeling**

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. ← software usually performs this step automatically

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.

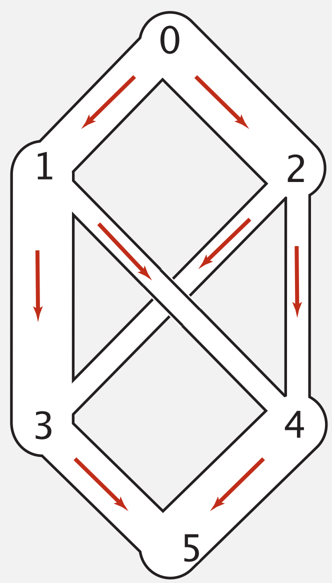
maxflow problem

V → 6

8 ← *E*

```
0 1  2.0
0 2  3.0
1 3  3.0
1 4  1.0
2 3  1.0
2 4  1.0
3 5  2.0
4 5  3.0
```

capacities



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.

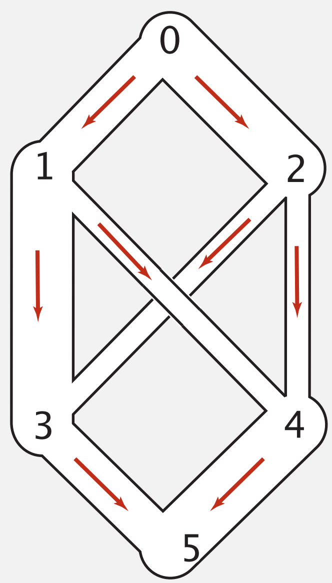
maxflow problem

V → 6

8 ← *E*

```
0 1  2.0
0 2  3.0
1 3  3.0
1 4  1.0
2 3  1.0
2 4  1.0
3 5  2.0
4 5  3.0
```

↑ *capacities*



LP formulation

Maximize $x_{35} + x_{45}$
subject to the constraints

$$\left.\begin{aligned} 0 \le x_{01} \le 2 \\ 0 \le x_{02} \le 3 \\ 0 \le x_{13} \le 3 \\ 0 \le x_{14} \le 1 \\ 0 \le x_{23} \le 1 \\ 0 \le x_{24} \le 1 \\ 0 \le x_{35} \le 2 \\ 0 \le x_{45} \le 3 \end{aligned}\right\} \text{capacity constraints}$$

$$\left.\begin{aligned} x_{01} = x_{13} + x_{14} \\ x_{02} = x_{23} + x_{24} \\ x_{13} + x_{23} = x_{35} \\ x_{14} + x_{24} = x_{45} \end{aligned}\right\} \text{flow conservation constraints}$$

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.

minimize $2z_{01} + 3z_{02} + 3z_{13} + z_{14} + z_{23} + z_{24} + 2z_{35} + 3z_{45}$

subject to the constraints

z_{01}	$\geq$	$y_0 - y_1$	z_{23}	$\geq$	$y_2 - y_3$
z_{02}	$\geq$	$y_0 - y_2$	z_{24}	$\geq$	$y_2 - y_4$
z_{13}	$\geq$	$y_1 - y_3$	z_{35}	$\geq$	$y_3 - y_5$
z_{14}	$\geq$	$y_1 - y_4$	z_{45}	$\geq$	$y_4 - y_5$
y_0	$=$	1	y_5	$=$	0
y_v	unrestricted		z_{vw}	$\geq$	0

source

sink

if $y_v = 1$ 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.

extreme point solution will be 0/1 (not always so lucky!)

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!

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.

  …

tractable

- Factoring
- NP-complete problems.

  …

intractable ?

see next lecture

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