http://algs4.cs.princeton.edu

LINEAR PROGRAMMING

- *brewer's problem*
- *simplex algorithm*
- *implementations*
- *reductions*

Linear programming

What is it? Problem-solving model for optimal allocation of scarce resources, among a number of competing activities that encompasses:

- Shortest paths, maxflow, MST, matching, assignment, ...
- $Ax = b$, 2-person zero-sum games, ...

can take an entire course on LP

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.

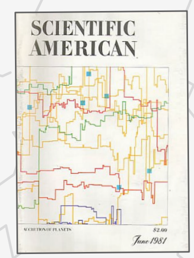
Sports. Scheduling ACC basketball, handicapping horse races.

http://algs4.cs.princeton.edu

Linear Programming

- *brewer's problem*
- *simplex algorithm*
- *implementations*
- *reductions*

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



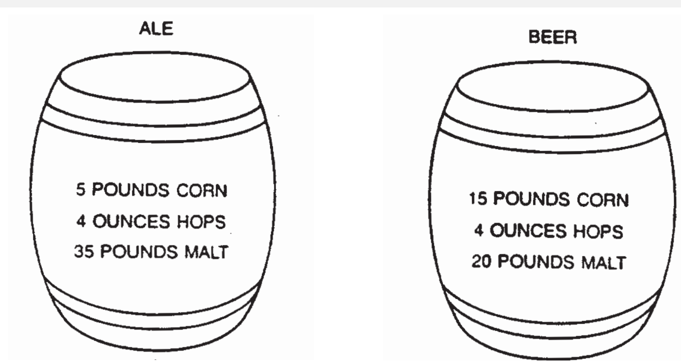
Toy LP example: brewer's problem

Small brewery produces ale and beer.

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

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**

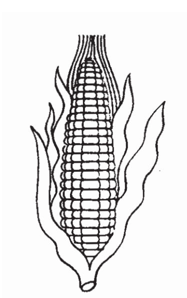
Toy LP example: brewer's problem

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

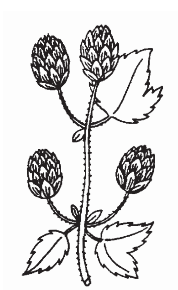
34 barrels × 35 lbs malt = 1190 lbs
[amount of available malt]

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 ?

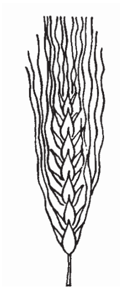
goods are divisible



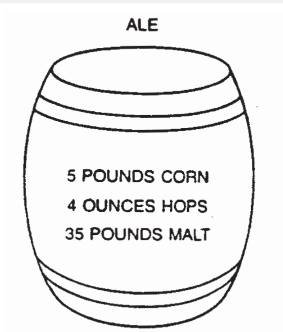
corn (480 lbs)



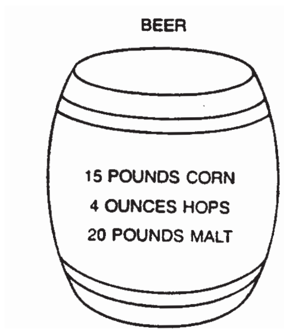
hops (160 oz)



malt (1190 lbs)



$13 profit per barrel



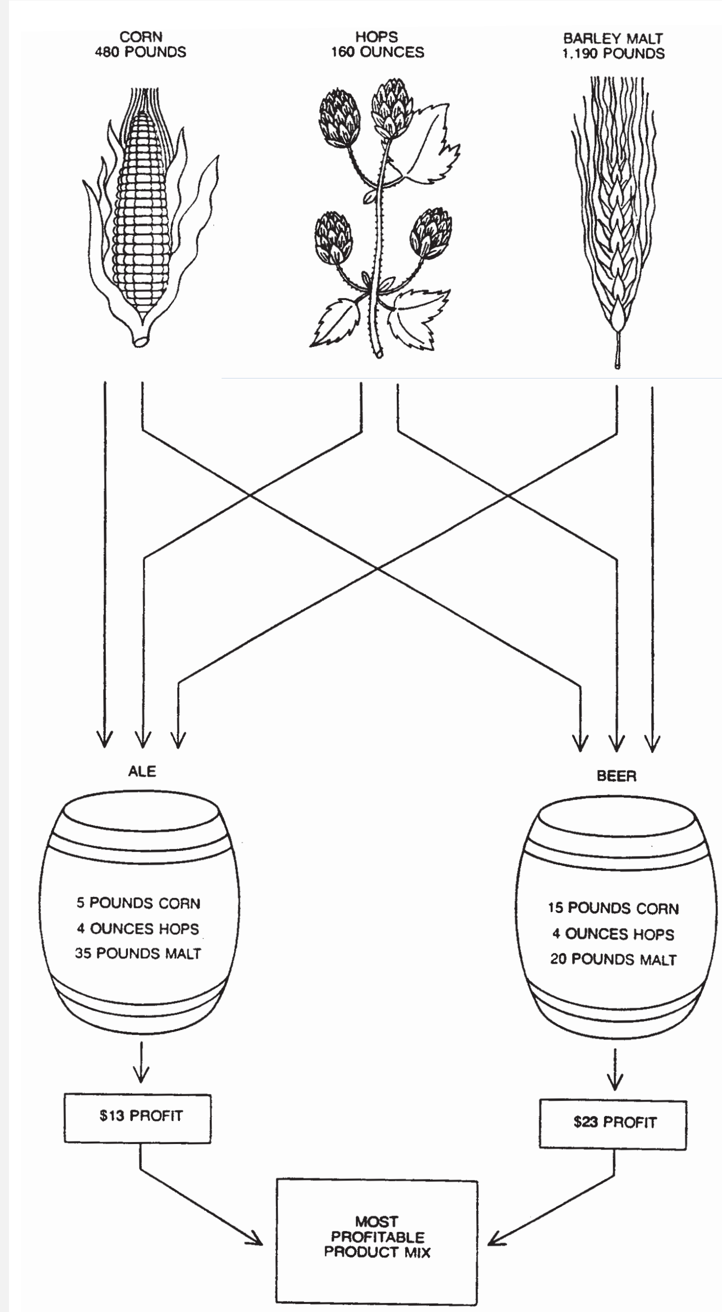
$23 profit per barrel

Brewer's problem: linear programming formulation

Linear programming formulation.

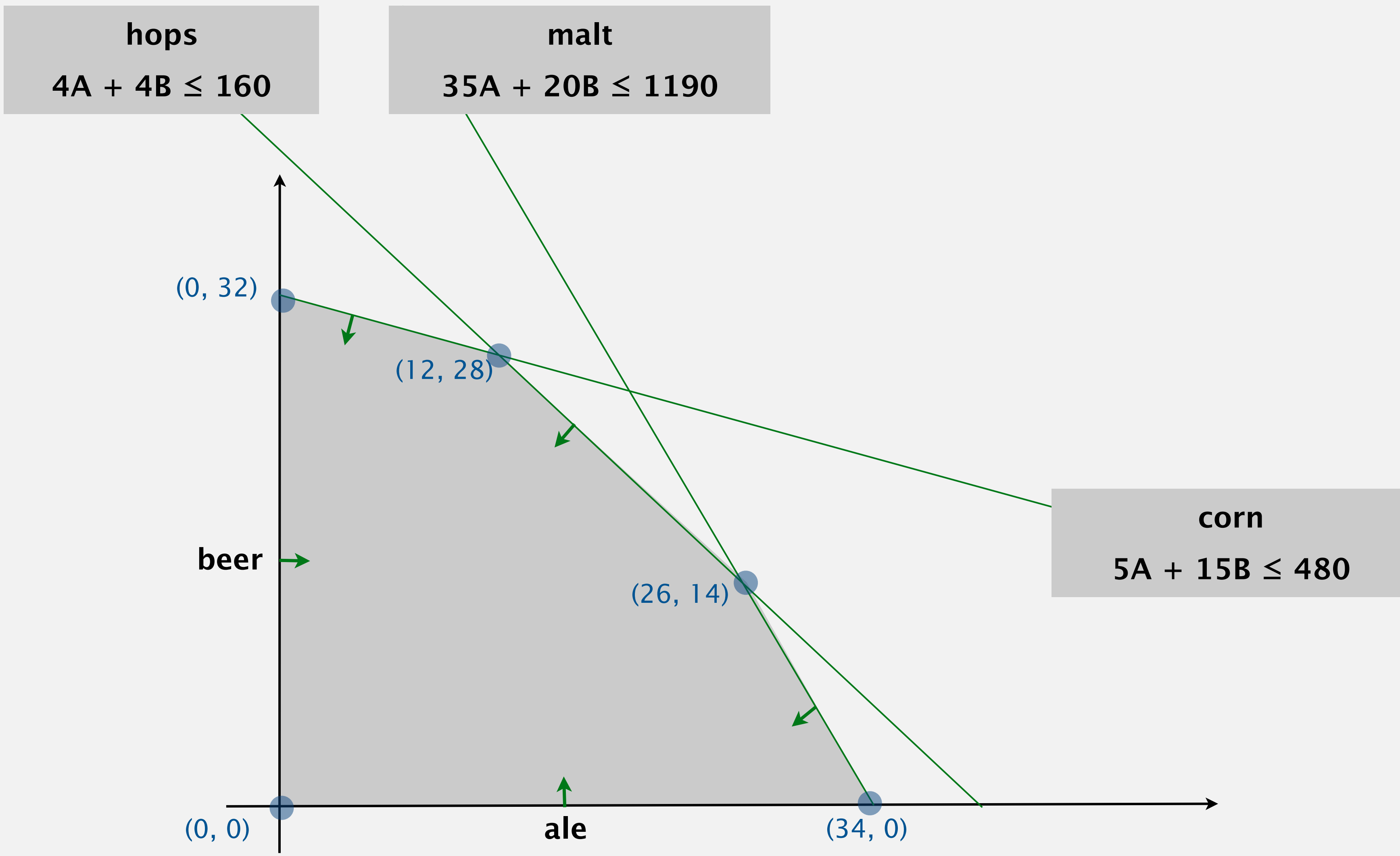
- Let A be the number of barrels of ale.
- Let B be the number of barrels of beer.

	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	

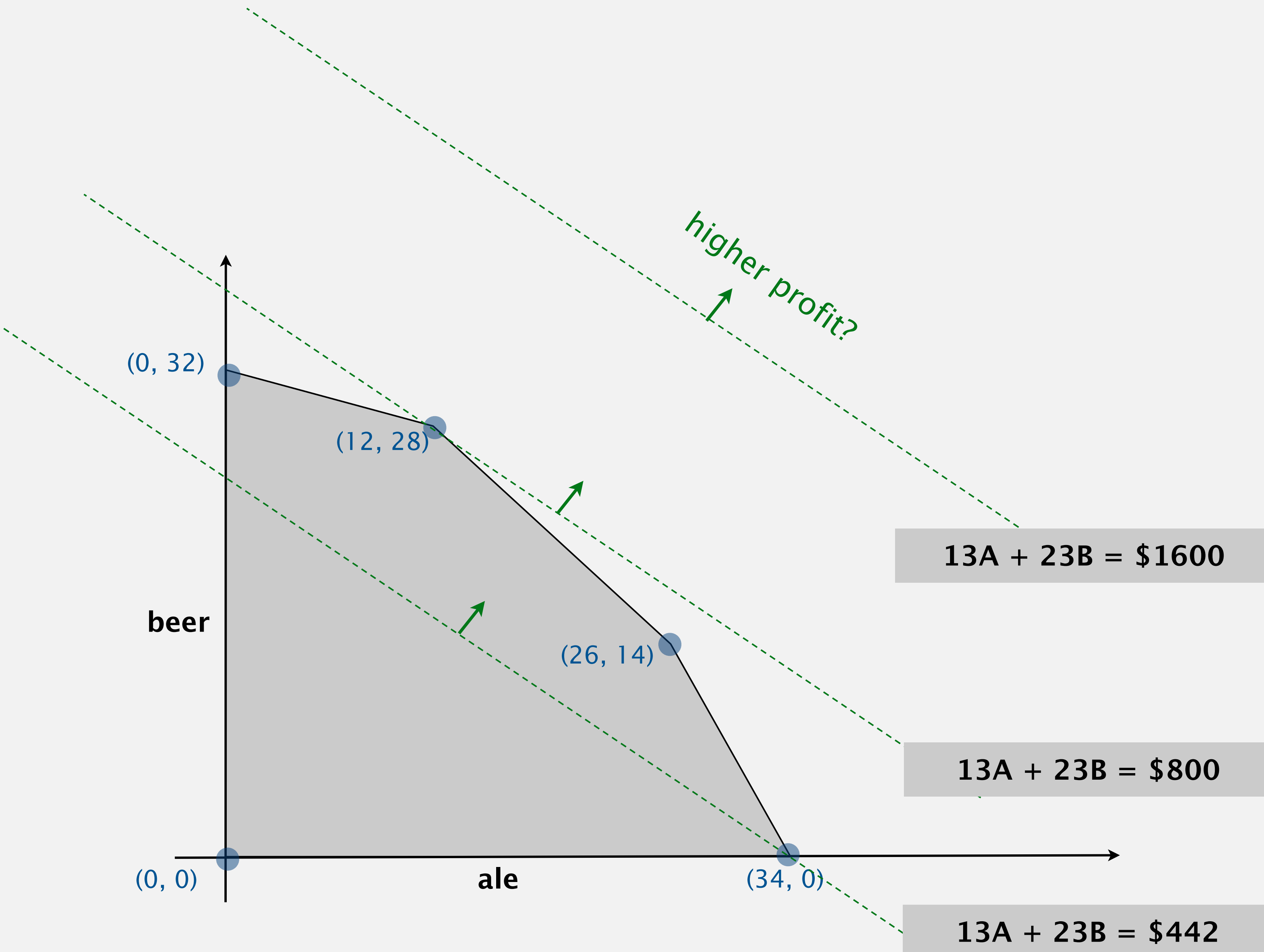


Brewer's problem: feasible region

Inequalities define halfplanes; feasible region is a convex polygon.



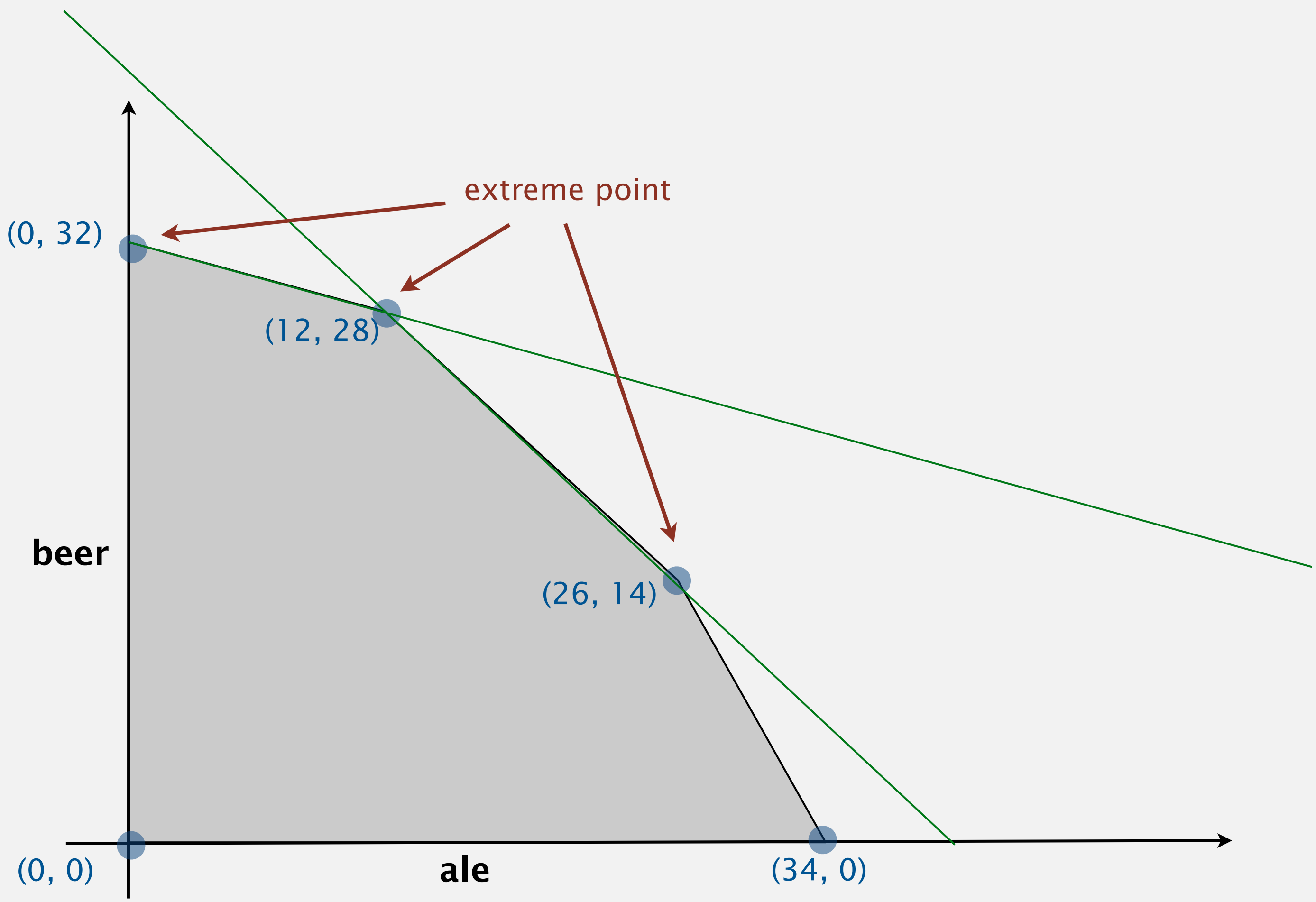
Brewer's problem: objective function



Brewer's problem: geometry

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.

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

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

primal problem (P)

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

matrix version

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

Caveat. No widely agreed notion of "standard form."

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.

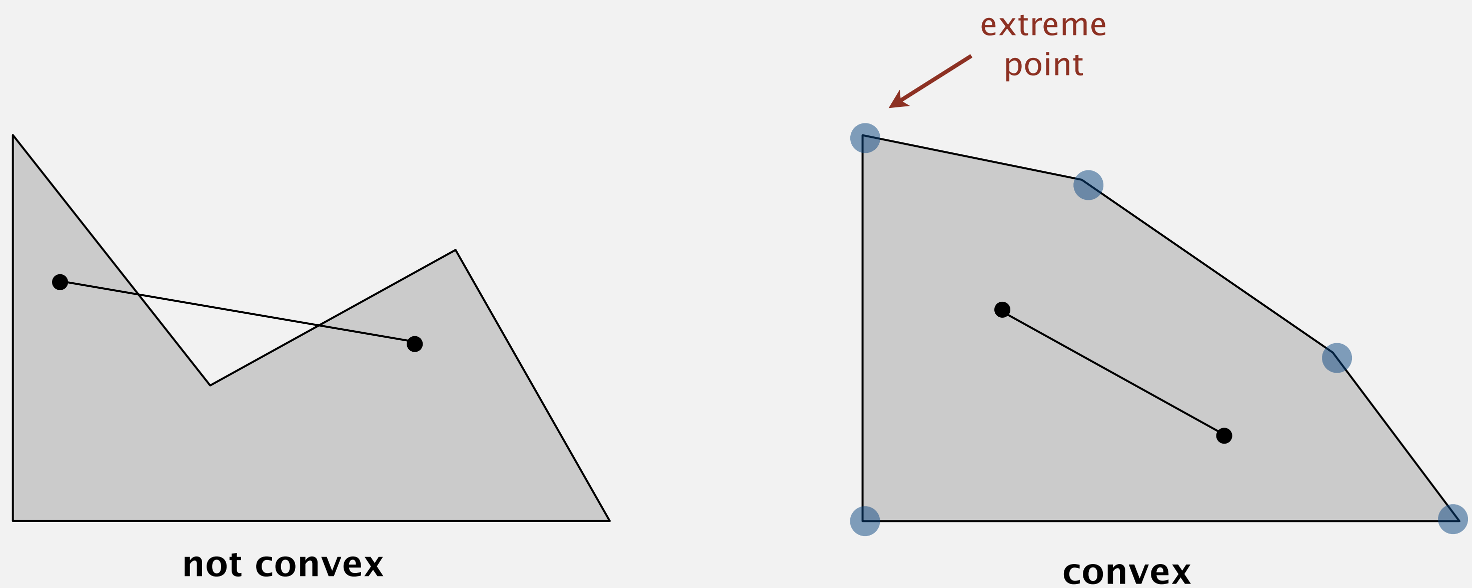
$$
\begin{array}{lrcrcccccccrcr}
\text{maximize} & Z & & & & & & & & & & & & \\
 & 13A & + & 23B & & & & & & & - & Z & = & 0 \\
\hline
\text{subject to the constraints} & 5A & + & 15B & + & S_C & & & & & & & = & 480 \\
 & 4A & + & 4B & & & + & S_H & & & & & = & 160 \\
 & 35A & + & 20B & & & & & + & S_M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

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.

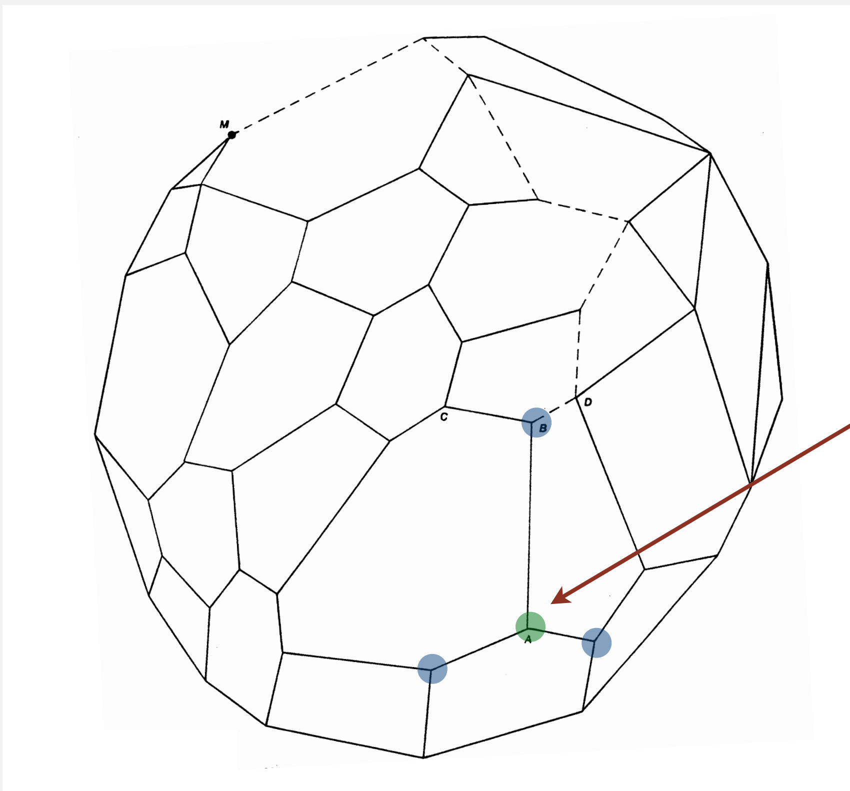


Warning. Don't always trust intuition in higher dimensions.

Geometry (continued)

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

- Good news: number of extreme points to consider is finite.
- Bad news : 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.

Algorithms

ROBERT SEDGEWICK | KEVIN WAYNE

http://algs4.cs.princeton.edu

LINEAR PROGRAMMING

- *brewer's problem*
- *simplex algorithm*
- *implementations*
- *reductions*

Linear Programming

- brewer's problem
- *simplex algorithm*
- implementations
- reductions

Algorithms

Robert Sedgewick | Kevin Wayne

http://algs4.cs.princeton.edu

Simplex algorithm

Simplex algorithm. [George Dantzig, 1947]

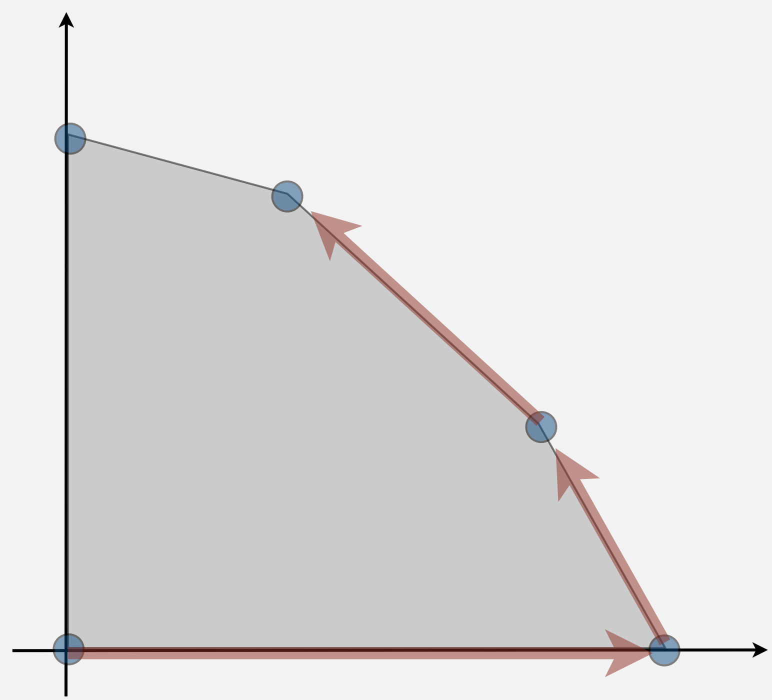
- Developed shortly after WWII in response to logistical problems, including Berlin airlift.
- Ranked as one of top 10 scientific algorithms of 20^{th} century.

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.



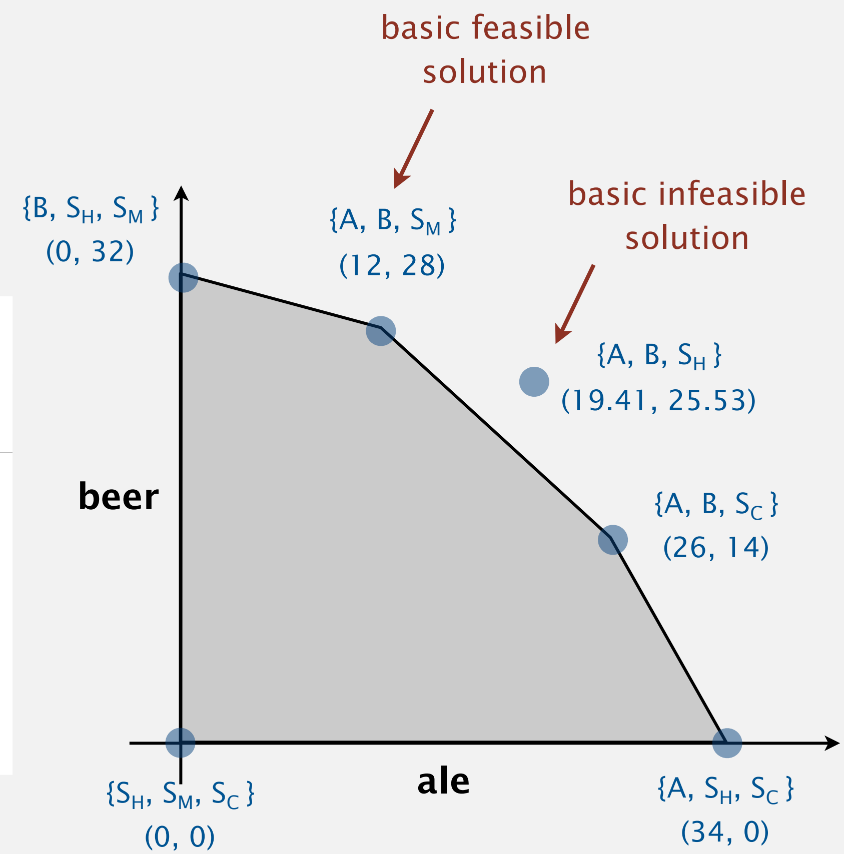
Simplex algorithm: basis

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		≥	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			$\geq$	0

basis = { S_C, S_H, S_M }

$A = B = 0$

$Z = 0$

$S_C = 480$

$S_H = 160$

$S_M = 1190$

one basic variable per row

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

no algebra needed

Simplex algorithm: pivot 1

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

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									
	$(16/3)\ A$			$-$	$(23/15)\ S_C$			$-\ Z$	$=$	-736
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

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

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

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

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

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?

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

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 $Z^* \le 800$ since $S_C,\, S_H \ge 0$.
- 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			$\ge$	0

basis = { A, B, S_M }

$S_C = S_H = 0$

$Z = 800$

$B = 28$

$A = 12$

$S_M = 110$

Algorithms

Robert Sedgewick | Kevin Wayne

http://algs4.cs.princeton.edu

Linear Programming

- *brewer's problem*
- *simplex algorithm*
- *implementations*
- *reductions*

Algorithms

Robert Sedgewick | Kevin Wayne

http://algs4.cs.princeton.edu

Linear Programming

- *brewer's problem*
- *simplex algorithm*
- *implementations*
- *reductions*

Simplex tableau

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

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

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

initial simplex tableaux

	n	m	1
m	A	I	b
1	c	0	0

Simplex tableau

Simplex algorithm transforms initial 2D array into solution.

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

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

m			x*
1	≤ 0	≤ 0	-Z*
	n	m	1

Simplex algorithm: initial simplex tableaux

Construct the initial 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;       // 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

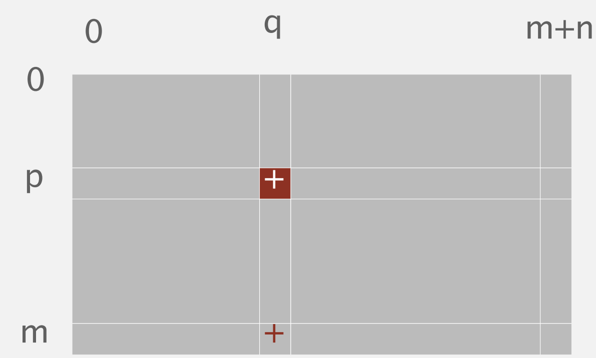
put I[][] into tableau

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][q] > 0) return q;

   return -1;
}
```

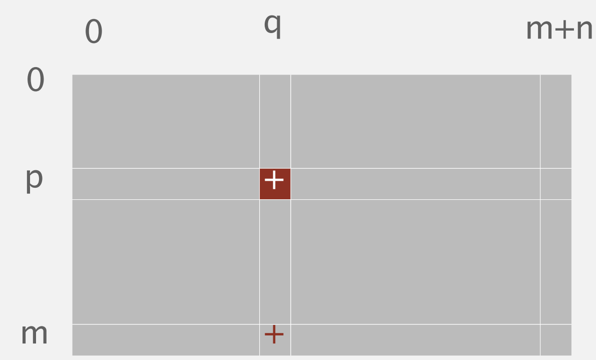
entering column q has positive objective function coefficient

optimal

Simplex algorithm: min-ratio rule

Find leaving row p using min ratio rule.

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




```
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;
}
```

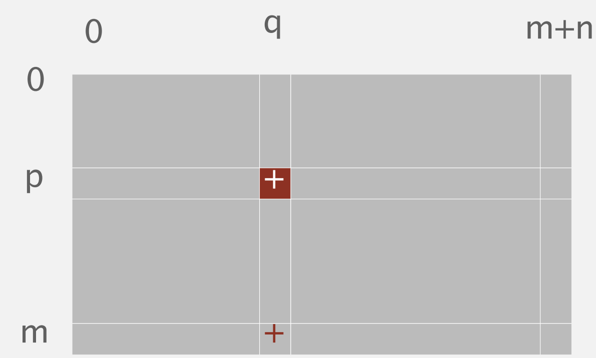
leaving row

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;
}
```

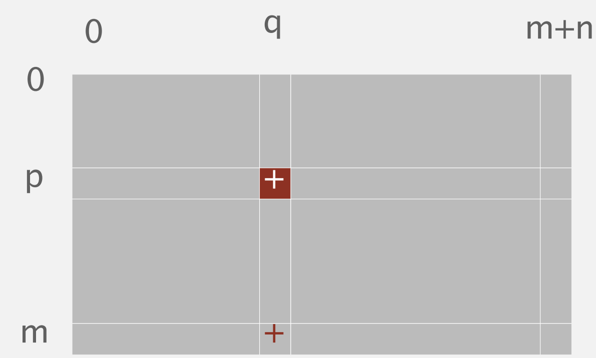
scale all entries but row p and column q

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

Simplex algorithm: running time

Remarkable property. In typical practical applications, simplex algorithm terminates after at most $2(m + n)$ pivots.

> *" Yes. Most of the time it solved problems with m equations in 2m or 3m steps— that was truly amazing. I certainly did not anticipate that it would turn out to be so terrific. I had had no experience at the time with problems in higher dimensions, and I didn't trust my geometrical intuition. For example, my intuition told me that the procedure would require too many steps wandering from one adjacent vertex to the next. In practice it takes few steps. In brief, one's intuition in higher dimensional space is not worth a damn! Only now, almost forty years from the time when the simplex method was first proposed, are people beginning to get some insight into why it works as well as it does. "*
>
> — *George Dantzig 1984*

Simplex algorithm: running time

Remarkable property. In typical practical applications, simplex algorithm terminates after at most $2(m+n)$ pivots.

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

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

Smoothed Analysis of Algorithms: Why the Simplex Algorithm Usually Takes Polynomial Time

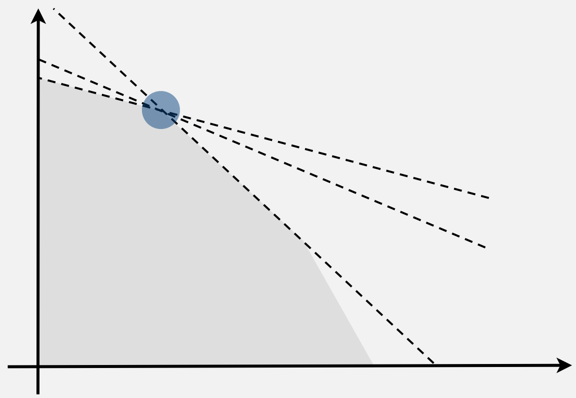
Daniel A. Spielman*
Department of Mathematics
M.I.T.
Cambridge, MA 02139
spielman@mit.edu

Shang-Hua Teng†
Akamai Technologies Inc. and
Department of Computer Science
University of Illinois at Urbana-Champaign
steng@cs.uiuc.edu

Simplex algorithm: degeneracy

Degeneracy. New basis, same extreme point.

"stalling" is common in practice



Cycling. Get stuck by cycling through different bases that all correspond to same extreme point.

- Doesn't occur in the wild.
- Bland's rule guarantees finite # of pivots.

choose lowest valid index for
entering and leaving columns

Simplex algorithm: implementation issues

To improve the bare-bones implementation.

- Avoid stalling. ← requires artful engineering
- Maintain sparsity. ← requires fancy data structures
- Numerical stability. ← requires advanced math
- Detect infeasibility. ← run "phase I" simplex algorithm
- Detect unboundedness. ← no leaving row

Best practice. Don't implement it yourself!

Basic implementations. Available in many programming environments.

Industrial-strength solvers. Routinely solve LPs with millions of variables.

Modeling languages. Simplify task of modeling problem as LP.

LP solvers: industrial strength

> “ *a benchmark production planning model solved using linear programming would have taken 82 years to solve in 1988, using the computers and the linear programming algorithms of the day. Fifteen years later—in 2003—this same model could be solved in roughly 1 minute, an improvement by a factor of roughly 43 million. Of this, a factor of roughly 1,000 was due to increased processor speed, whereas a factor of roughly 43,000 was due to improvements in algorithms!* ”
>
> — *Designing a Digital Future*
>
> *(Report to the President and Congress, 2010)*

Brief history

1939. Production, planning. [Kantorovich]

1947. Simplex algorithm. [Dantzig]

1947. Duality. [von Neumann, Dantzig, Gale-Kuhn-Tucker]

1947. Equilibrium theory. [Koopmans]

1948. Berlin airlift. [Dantzig]

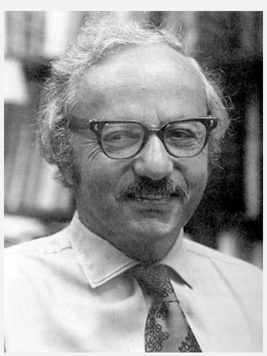
1975. Nobel Prize in Economics. [Kantorovich and Koopmans]

1979. Ellipsoid algorithm. [Khachiyan]

1984. Projective-scaling algorithm. [Karmarkar]

1990. Interior-point methods. [Nesterov-Nemirovskii, Mehorta, ...]

Kantorovich



George Dantzig

von Neumann

Koopmans

Khachiyan

Karmarkar

Algorithms

Robert Sedgewick | Kevin Wayne

http://algs4.cs.princeton.edu

Linear Programming

- brewer's problem
- simplex algorithm
- **implementations**
- reductions

Algorithms

Robert Sedgewick | Kevin Wayne

http://algs4.cs.princeton.edu

Linear Programming

- *brewer's problem*
- *simplex algorithm*
- *implementations*
- *reductions*

Reductions to standard form

Minimization problem. Replace $min\ 13A + 15B$ with $max - 13A - 15B$.

≥ constraints. Replace $4A + 4B \geq 160$ with $4A + 4B - S_H = 160$, $S_H \geq 0$.

Unrestricted variables. Replace B with $B = B_0 - B_1$, $B_0 \geq 0$, $B_1 \geq 0$.

nonstandard form

$$\begin{array}{lrcrcl} \textit{minimize} & 13A & + & 15B & & \\ \textit{subject to:} & 5A & + & 15B & \leq & 480 \\ & 4A & + & 4B & \geq & 160 \\ & 35A & + & 20B & = & 1190 \\ & A & & & \geq & 0 \\ & & & B & \text{is} & \text{unrestricted} \end{array}$$

standard form

$$\begin{array}{lrcrcrcrcrcr} \textit{maximize} & -13A & - & 15B_0 & + & 15B_1 & & & & & & \\ \textit{subject to:} & 5A & + & 15B_0 & - & 15B_1 & + & S_C & & & = & 480 \\ & 4A & + & 4B_0 & - & 4B_1 & & & - & S_H & = & 160 \\ & 35A & + & 20B_0 & - & 20B_1 & & & & & = & 1190 \\ & A & & B_0 & & B_1 & & S_C & & S_H & \geq & 0 \end{array}$$

Modeling

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

- Process of formulating an LP model for a problem.
- Solution to LP for a specific problem gives solution to the problem.

1. Identify variables.
2. Define constraints (inequalities and equations).
3. Define objective function.
4. Convert to standard form. ← software usually performs this step automatically

Examples.

- Maxflow.
- Shortest paths.
- Bipartite matching.
- Assignment problem.
- 2-person zero-sum games.

...

Maxflow problem (revisited)

Input. Weighted digraph G, single source s and single sink t.

Goal. Find maximum flow from s to t.

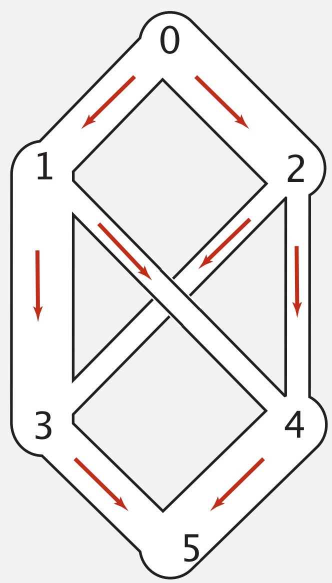
maxflow problem

V → 6

E → 8

```
6
8
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.

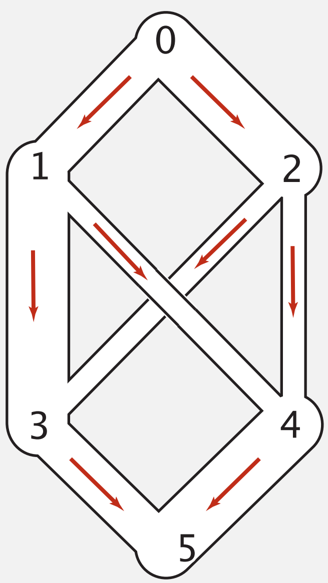
maxflow problem

V

E

```
6
8
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

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

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

Maximum cardinality bipartite matching problem

Input. Bipartite graph.

Goal. Find a matching of maximum cardinality.

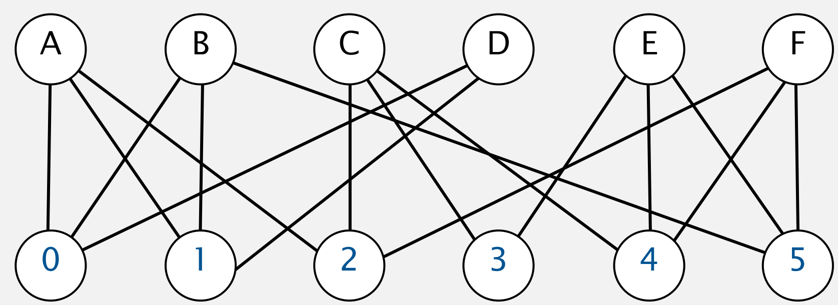
set of edges with no vertex appearing twice

Interpretation. Mutual preference constraints.

- People to jobs.
- Students to writing seminars.

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

Example: job offers



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

Maximum cardinality bipartite matching problem

LP formulation. One variable per pair.

Interpretation. $x_{ij} = 1$ if person i assigned to job j.

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

at most one job per person	at most one person per job
$x_{A0} + x_{A1} + x_{A2} \le 1$	$x_{A0} + x_{B0} + x_{D0} \le 1$
$x_{B0} + x_{B1} + x_{B5} \le 1$	$x_{A1} + x_{B1} + x_{D1} \le 1$
$x_{C2} + x_{C3} + x_{C4} \le 1$	$x_{A2} + x_{C2} + x_{F2} \le 1$
$x_{D0} + x_{D1} \le 1$	$x_{C3} + x_{E3} \le 1$
$x_{E3} + x_{E4} + x_{E5} \le 1$	$x_{C4} + x_{E4} + x_{F4} \le 1$
$x_{F2} + x_{F4} + x_{F5} \le 1$	$x_{B5} + x_{E5} + x_{F5} \le 1$

$$\text{all } x_{ij} \ge 0$$

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

Corollary. Can solve matching problem by solving LP. ← not usually so lucky!

Linear programming perspective

Q. Got an optimization problem?

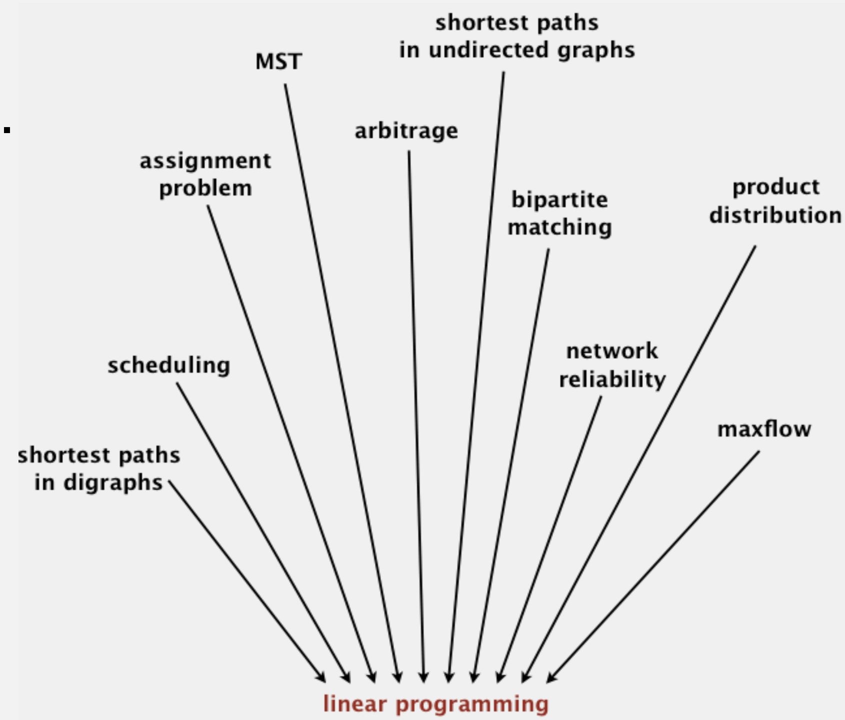
Ex. Maxflow, bipartite matching, shortest paths, … [many, many, more]

Approach 1: Use a specialized algorithm to solve it.

- Algorithms 4/e.
- Vast literature on algorithms.

Approach 2: Use linear programming.

- Many problems are easily modeled as LPs.
- Commercial solvers can solve those LPs.
- Might be slower than specialized solution (but you might not care).



Got an LP solver? Learn to use it!

Universal problem-solving model (in theory)

Is there a universal problem-solving model?

- Maxflow.
- Shortest paths.
- 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.

LINEAR PROGRAMMING

- *brewer's problem*
- *simplex algorithm*
- *implementations*
- *reductions*

Algorithms

ROBERT SEDGEWICK | KEVIN WAYNE

http://algs4.cs.princeton.edu

http://algs4.cs.princeton.edu

Linear Programming

- *brewer's problem*
- *simplex algorithm*
- *implementations*
- *reductions*