Required reading is from Chvátal. Optional reading is from these reference books:

A: AMPL
K: Karloff
M: Murty's LP book
MC: Murty's LP/CO book
S: Schrijver
V: Vanderbei
Vz: *Approximation Algorithms* by Vijay Vazirani, Springer 2001.
WV: Winston & Venkataramanan

Unit 4: Efficient Implementation of Simplex Algorithm

23	97-100	Matrix Representations of LPs
24	100-105	Revised Simplex Algorithm: High Level
25	Ch.6	Review of Gaussian Elimination
26	105-111	Solving Eta Systems
		More Applications
27	195-200, 207-211	The Cutting Stock Problem
28	201-207	Branch-and-Bound Algorithms

Unit 5: Extensions of Theory and Algorithms

29	118-119	General Form LPs
30	119-129, 132-133	Upper Bounding
31	130-132, 133-134 242-243	Generalized Fundamental Theorem
32	143-146	Inconsistent Systems of Linear Inequalities
33	Ex.16.10	Theorems of Alternatives
34	152-157	Dual Simplex Algorithm
35	158-162	Sensitivity Analysis
36	162-166	Parametric LPs
		More Applications
37	(WV 9.8)	Cutting Planes for ILP
38	262-269	Applications to Geometry

Unit 6: Network Algorithms

39	291-295	The Transshipment Problem
40	296-303	Network Simplex Algorithm

Unit 7: Polynomial-Time LP

41	443-452, (K Ch.4)	Overview of the Ellipsoid Algorithm

Unit 8: Beyond Linearity

42	(MC16.4.4V 23.1)	Quadratic Programming Examples
43	(MC, 16.4.4)	Solving Quadratic Programs
		More Applications
44	(Vz, Ch.26)	Semidefinite Programming: Approximating MAX CUT

Course Fact Sheet for CSCI 5654: Linear Programming

Time	M W 4:00 – 5:15 MUEN E118 or ECCS 1B12 (CAETE)
Instructor	Hal Gabow Office ECOT 624 Mailbox ECOT 717 Office Phone 492-6862 Email `hal@cs.colorado.edu`
Office Hours	M 2:00–3:50, 5:15 –6:00; W 3:00–3:50; or by appointment These times may change slightly– see my home page for the authoritative list. After class is always good for questions.
Grader	TBA
Prerequisites	Undergraduate courses in linear algebra & data structures e.g., MATH 3130 & CSCI 2270, or their equivalents
Requirements	Weeks 1–9: Written problem sets, sometimes using LP solvers Takehome exam, worth 2 HWs Weeks 10–14: multipart reading project Week 15–16: Takehome final, worth 3–3.5 HWs. Due Fri. Dec. 14 (day before official final) Final grades are assigned on a curve. In Fall '05 the grades were: Campus: 11 As 2 Bs 2 Cs 1 F CATECS: 1 A 1 B
Pretaped class	Mon. Oct. 22 (*FOCS* Conference). The pretaping is Wed. Oct. 17, 5:30-6:45, same room. Additional cancellations/pretaping sessions possible.

Homework

Most assignments will be 1 week long, due on a Wednesday. That Wednesday the current assignment is turned in, a solution sheet is distributed in class & the next assignment is posted on the class website. The due date for CAETE students *who do not attend live class* is 1 week later (see the email message to CAETE students). A few assignments may be $1\frac{1}{2} - 2$ weeks & the schedule will be modified accordingly.

Writeups must be legible. Computer-typeset writeups are great but are not required. If you're handwriting an assignment use lined paper and leave lots of space (for legibility, and comments by the grader). Illegible homework may be returned with *no credit*.

Grading: Each assignment is worth 100 points. Some assignments will have an extra credit problem worth 20 points. The extra credit is more difficult than the main assignment and is meant for students who want to go deeper into the subject. Extra credit points are recorded separately as "HEC"s. Final grades are based on the main assignments and are only marginally influenced by HECs or other ECs.
In each class you can get 1 "VEC" (verbal extra credit) for verbal participation.

Collaboration Policy: Homework should be your own. There will be no need for collaborative problem solving in this course. Note this policy is different from CSCI 5454. If there's any doubt ask me. You will get a 0 for copying even part of an assignment from any source; a second violation will result in failing the course.

Anyone using any of my solution sheets to do homework will receive an *automatic F* in the course. (One student wound up in jail for doing this.)

All students are required to know and obey the University of Colorado Student Honor Code, posted at `http://www.colorado.edu/academics/honorcode`. The class website has this link posted under Campus Rules, which also has links to policies on classroom behavior, religious observances and student disability services. Each assignment must include a statement that the honor code was obeyed – see directions on the class website. I used to lower grades because of honesty issues, but the Honor Code is working and I haven't had to do this in a while.

Late homeworks: Homeworks are due in class on the due date. Late submissions will not be accepted. Exceptions will be made only if arrangements are made with me 1 week in advance. When this is impossible (e.g., medical reasons) documentation will be required (e.g., physician's note).

Ignorance of these rules is not an excuse.

Website & Email

The class website is `http://www.cs.colorado.edu/~hal/CS5654/home.html`. It is the main form of communication, aside from class. Assignments, current HW point totals and other course materials will be posted there.

You can email me questions on homework assignments. I'll post any needed clarifications on the class website. I'll send email to the class indicating new information on the website, e.g., clarifications of the homework assignments, missing office hours, etc. Probably my email to the class will be limited to pointers to the website. I check my email until 9:30 PM most nights.

Inappropriate email: Email is great for clarifying homework assignments. I try to answer all email questions. But sometimes students misuse email and ask me to basically do their work for them. Don't do this.

Text *Linear Programming*
by Vašek Chvátal, W.H. Freeman and Co., New York, 1984

References On reserve in Lester Math-Physics Library, or available from me.

Background in Linear Algebra:

Linear Algebra and its Applications
by G. Strang, Harcourt College Publishers, 1988 (3rd Edition)

Similar to Chvátal:

Linear Programming: Foundations and Extensions
by Robert J. Vanderbei, Kluwer Academic Publishers, 2001 (2nd Edition)
(optional 2nd Text)

Linear Programming
by K.G. Murty, Wiley & Sons, 1983 (revised)

Linear and Combinatorial Programming
by K.G. Murty, Wiley & Sons, 1976

Introduction to Mathematical Programming, 4th Edition
by W.L. Winston, M. Venkataramanan, Thomson, 2003

Linear Programming
by H. Karloff, Birkhäuser, 1991

More Theoretical:

Theory of Linear and Integer Programming
by A. Schrijver, John Wiley, 1986

Integer and Combinatorial Programming
by G.L. Nemhauser and L.A. Wolsey, John Wiley, 1988

Geometric Algorithms and Combinatorial Optimization
by M. Grötschel, L. Lovász and A. Schrijver, Springer-Verlag, 1988

Using LP:

AMPL: A Modeling Language for Mathematical Programming
by R.Fourer, D.M.Gay, and B.W.Kernighan, Boyd & Fraser, 1993

Course Goals

Linear Programming is one of the most successful models for optimization, in terms of both real-world computing and theoretical applications to CS, mathematics, economics & operations research. This course is an introduction to the major techniques for LP and the related theory, as well as touching on some interesting applications.

Course Content

A course outline is given by the Handout List (Handout #i). We follow Chvátal's excellent development, until the last two units on extra material.

Unit 1 is an *Overview*. It defines the LP problem & illustrates LP models.

Unit 2, *Fundamentals*, covers the simplex method at a high level. Simplex is used in most codes for solving real-world LPs. Our conceptual treatment culminates in proving the Fundamental Theorem of LP, which summarizes what Simplex does.

Unit 3 gives the basics of one of the most important themes in operations research and theoretical computer science, *Duality*. This theory is the basis of other LP methods as well as many combinatorial algorithms. We touch on applications to economics and game theory.

Unit 4 returns to Simplex to present its *Efficient Implementation*. These details are crucial for realizing the speed of the algorithm. We cover the technique of delayed column generation and its application to industrial problems such as cutting stock, as well as the branch-and-bound method for integer linear programming.

Unit 5, *Extensions*, gives other implementation techniques such as upper-bounding and sensitivity analysis, as well as some general theory about linear inequalities, and the Dual Simplex Algorithm, which recent work indicates is more powerful than previously thought. We introduce the cutting plane technique for Integer Linear Programming.

Unit 6 is an introduction to *Network Algorithms*. These special LPs, defined on graphs, arise often in real-life applications. We study the Network Simplex Algorithm, which takes advantage of the graph structure to gain even more efficiency.

Unit 7, *Polynomial-Time Linear Programming*, surveys the ellipsoid method. This is a powerful tool in theoretical algorithm design, and it opens the door to nonlinear programming. The Supplemental Material covers Karmarkar's algorithm, an alternative to Simplex that is often even more efficient.

Unit 8 is a brief introduction to nonlinear methods: quadratic programming (& the Markowitz model for assessing risk versus return on financial investments) and its generalization to semidefinite programming. We illustrate the latter with the groundbreaking Goemans-Williamson SDP algorithm for approximating the maximum cut problem in graphs.

The *Supplemental Material* in Unit 9 will be covered as time permits.

Linear Algebra

Chvátal's treatment centers on the intuitive notion of *dictionary*. Units 1–3 use very little linear algebra, although it's still helpful for your intuition. Units 4–6 switch to the language of linear algebra, but we really only use very big ideas. Units 7–8 use more of what you learned at the start of an introductory linear algebra course. The course is essentially self-contained as far as background from linear algebra is concerned.

Tips on using these notes

Class lectures will work through the notes. This will save you writing.

Add comments at appropriate places in the notes. Use backs of pages for notes on more extensive discussions (or if you like to write big!). If I switch to free sheets of paper for extended discussions not in the notes, you may also want to use free sheets of paper.

A multi-colored pen can be useful (to separate different comments; for complicated pictures; to color code remarks, e.g. red = "important," etc.) Such a pen is a crucial tool for most mathematicians and theoretical computer scientists!

If you want to review the comments I wrote in class, I can reproduce them for you.

The notes complement the textbook. The material in Chvátal corresponding to a handout is given in the upper left corner of the handout's page 1, & in Handout #i. The notes follow Chvátal, but provide more formal discussions of many concepts as well as alternate explanations & additional material. The notes are succinct and require additional explanation, which we do in class. You should understand both Chvátal and my notes.

Extra credit for finding a mistake in these notes.

What you didn't learn in high school algebra

High School Exercise. Solve this system of inequalities (*Hint*: It's inconsistent!):

$$\begin{aligned} 4x+2y &\geq 8 \\ 3x+4y &\geq 12 \\ 3x+2y &\leq 7 \end{aligned}$$

systems of inequalities are much harder to solve than systems of equations!

linear programming gives the theory and methods to solve these systems
in fact LP is mathematically equivalent to solving these systems

A "real-world" LP

Power Flakes' new product will be a mixture of corn & oats.
1 serving must supply at least 8 grams of protein, 12 grams of carbohydrates, & 3 grams of fat.
1 ounce of corn supplies 4,3, and 2 grams of these nutrients, respectively.
1 ounce of oats supplies 2,4, and 1 grams, respectively.
Corn can be bought for 6 cents an ounce, oats at 4 cents an ounce.
What blend of cereal is best?
"Best" means that it minimizes the cost – ignore the taste!

LP Formulation
x = # ounces of corn per serving; y = # ounces of oats per serving

$$\begin{aligned} \text{minimize } z = {} & 6x+4y \\ \text{subject to } & 4x+2y \geq 8 \\ & 3x+4y \geq 12 \\ & 2x+\ y \geq 3 \\ & \quad x, y \geq 0 \end{aligned}$$

assumptions for our model:
linearity – proportionality, additivity
(versus diminishing returns to scale, economies of scale, protein complementarity)
continuity – versus integrality

Linear Programming – optimize a linear function subject to linear constraints

Standard Form (changes from text to text)

objective function

$$
\begin{array}{lrll}
\text{maximize } z = & \sum_{j=1}^{n} c_j x_j & & \\
\text{subject to} & \sum_{j=1}^{n} a_{ij} x_j & \le b_i & (i = 1, \ldots, m) \\
 & x_j & \ge 0 & (j = 1, \ldots, n)
\end{array}
\Bigg\} \longleftarrow \textit{constraints}
$$

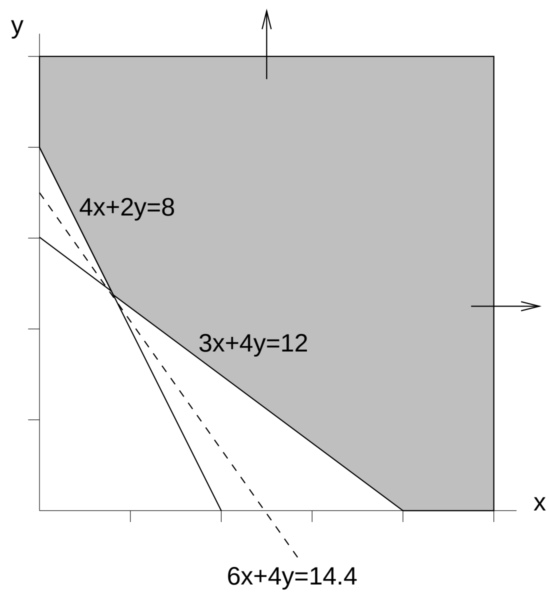
nonnegativity constraints

cost coefficient c_j, *decision variable* x_j, *right-hand side coefficient* b_i

a *feasible solution* satisfies all the constraints
an *optimum solution* is feasible and achieves the optimum objective value

Activity Space Representation

n dimensions, x_j = level of activity j



Requirement Space Representation (m dimensions – see Handout#33, p.2)

Real-world LPs

blending problem – find a least cost blend satisfying various requirements
e.g., gasoline blends (Chvátal, p.10, ex.1.7); diet problem (Chvátal, pp.3–5); animal feed; steel composition

resource allocation problem – allocate limited *resources* to various *activities* to maximize profit
e.g., forestry (Chvátal, pp.171–6); Chvátal, p.10, ex.1.6; beer production

multiperiod scheduling problems – schedule activities in various time periods to maximize profit
e.g., Chvátal, p.11, ex.1.8; cash flow; inventory management; electric power generation

And Solving Them

LP-solvers include `CPLEX` (the industrial standard), `MINOS`, `MOSEK`, `LINDO`
most of these have free (lower-powered) versions as downloads

modelling languages facilitate specification of large LPs
e.g., here's an `AMPL` program for a resource-allocation problem (from *AMPL* text):

the model is in a `.mod` file:

```
set PROD;   # products
set STAGE;  # stages
param rate {PROD,STAGE} > 0; # tons per hour in each stage
param avail {STAGE} >= 0;    # hours available/week in each stage
param profit {PROD};         # profit per ton
param commit {PROD} >= 0;    # lower limit on tons sold in week
param market {PROD} >= 0;    # upper limit on tons sold in week
var Make {p in PROD} >= commit[p], <= market[p]; # tons produced
maximize total_profit: sum {p in PROD} profit[p] * Make[p];
subject to Time {s in STAGE}:
   sum {p in PROD} (1/rate[p,s]) * Make[p] <= avail[s];
   # In each stage: total of hours used by all products may not exceed hours available
```

& the data is in a `.dat` file:

```
set PROD := bands coils plate;
set STAGE := reheat roll;
param rate:  reheat  roll :=
  bands        200    200
  coils        200    140
  plate        200    160 ;
param:    profit  commit  market :=
  bands     25     1000    6000
  coils     30      500    4000
  plate     29      750    3500 ;
param avail :=  reheat 35   roll   40 ;
```

LP versus ILP

an *Integer Linear Program (ILP)* is the same as an LP
but the variables x_j are required to be integers
much harder!

Two Examples

Knapsack Problem

$$\begin{aligned} &\text{maximize } z = 3x_1 + 5x_2 + 6x_3 \\ &\text{subject to } x_1 + 2x_2 + 3x_3 \le 5 \\ &\qquad\qquad 0 \le x_i \le 1, \quad x_i \text{ integral} \qquad i = 1,2,3 \end{aligned}$$

this is a "knapsack problem":
items 1,2 & 3 weigh 1,2 & 3 pounds respectively
and are worth \$3,\$5 & \$6 respectively
put the most valuable collection of items into a knapsack that can hold 5 pounds

the corresponding LP (i.e., drop integrality constraint) is easy to solve by the *greedy method*–
add items to the knapsack in order of decreasing value per pound
if the last item overflows the knapsack, add just enough to fill it
we get $x_1 = x_2 = 1, x_3 = 2/3$, $z = 12$

the best integral solution is $x_1 = 0, x_2 = x_3 = 1$, $z = 11$

but the greedy method won't solve arbitrary LPs!

Exercise. The example LP (i.e., continuous knapsack problem) can be put into a form where the greedy algorithm is even more obvious, by rescaling the variables. (a) Let y_i be the number of pounds of item i in the knapsack. Show the example LP is equivalent to the LP $(*)$ below. (b) Explain why it's obvious that the greedy method works on $(*)$.

$$(*) \qquad \begin{aligned} &\text{maximize } z = 3y_1 + \tfrac{5}{2}y_2 + 2y_3 \\ &\text{subject to } y_1 + y_2 + y_3 \le 5 \\ &\qquad\qquad 0 \le y_i \le i, \qquad i = 1,2,3 \end{aligned}$$

$(*)$ is a special case of a *polymatroid* – a broad class of LP's with 0/1 coefficients where the greedy method works correctly.

Traveling Salesman Problem (TSP)

TSP is the problem of finding the shortest cyclic route through n cities,
visiting each city exactly once
starting & ending at the same city

index the cities from 1 to n
let c_{ij}, $i, j = 1, \ldots, n$ denote the direct distance between cities i & j
we'll consider the *symmetric TSP*, where $c_{ij} = c_{ji}$

for $1 \le i < j \le n$,
let $x_{ij} = 1$ if cities i & j are adjacent in the tour, otherwise $x_{ij} = 0$

symmetric TSP is this ILP:

$$
\begin{array}{llllll}
\text{minimize } z = & \sum_{i<j} c_{ij} x_{ij} & & & & \\
\text{subject to} & \sum_{k \in \{i,j\}} x_{ij} & = & 2 & k = 1, \ldots, n & \text{(enter \& leave each city)} \\
 & \sum_{|\{i,j\} \cap S| = 1} x_{ij} & \ge & 2 & \emptyset \subset S \subset \{1, \ldots, n\} & \text{(``subtour elimination constraints'')} \\
 & x_{ij} & \in & \{0,1\} & 1 \le i < j \le n &
\end{array}
$$

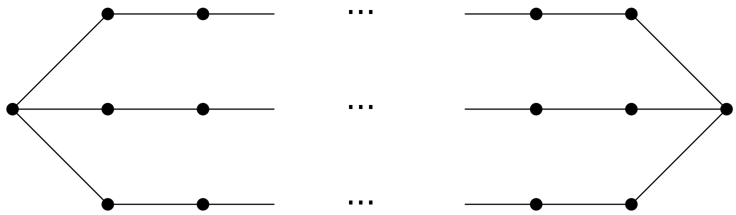
we form the *(Held-Karp) LP relaxation* of this ILP by dropping the integrality constraints
i.e., replace the last line by $x_{ij} \ge 0$, $1 \le i < j \le n$

let z_{ILP} and z_{LP} denote the optimum objective values of the 2 problems
assume distances satisfy the triangle inequality
$c_{ij} + c_{jk} \ge c_{ik}$ for all i, j, k
then we know that $z_{ILP} \le (3/2) z_{LP}$, i.e., the *integrality gap* is $\le 3/2$

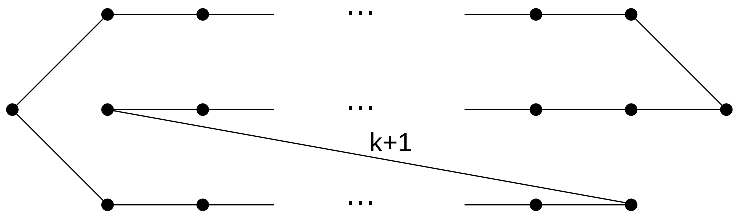
experimentally the Held-Karp lower bound is typically above $.99 z_{ILP}$

the 4/3 *Conjecture* states (unsurprisingly) that the integrality gap is always $\le 4/3$

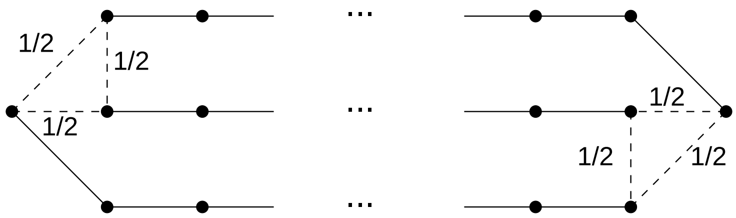
Exercise. We'll show the integrality gap can be $\geq 4/3 - \epsilon$, for any $\epsilon > 0$. Here's the graph:



$3k + 2$ vertices; k vertices are in the top horizontal path. For any 2 vertices i, j, c_{ij} is the number of edges in a shortest path from i to j.



A tour with length $4k + 2$.



A fractional solution of length $3k + 3$
$\big(3k + 3 = 3(k - 1) + 2(1 + 1/2 + 1/2 + 1/2(2))\big)$.

(a) Explain why the above fractional solution is feasible. *Hint*: Concentrate on the 3 paths of solid edges. (b) Explain why any fractional solution has length $\geq 3k + 2$. (c) Explain why any tour has length $\geq 4k + 2$. *Hint*: Concentrate on the length 1 edges traversed by the tour. They break up into subpaths beginning and ending at 1 of the 2 extreme points of the graph. (d) Conclude that for this example, $\lim_{k\to\infty} z_{ILP}/z_{LP} = 4/3$.

a *linear program* is a problem that can be put into standard (maximization) form –

$$\text{maximize } z = \sum_{j=1}^{n} c_j x_j$$

$$\begin{array}{llll} \text{subject to} & \sum_{j=1}^{n} a_{ij} x_j & \le b_i & (i = 1, \ldots, m) \\ & x_j & \ge 0 & (j = 1, \ldots, n) \end{array}$$

Standard minimization form

$$\text{minimize } z = \sum_{j=1}^{n} c_j x_j$$

$$\begin{array}{llll} \text{subject to} & \sum_{j=1}^{n} a_{ij} x_j & \ge b_i & (i = 1, \ldots, m) \\ & x_j & \ge 0 & (j = 1, \ldots, n) \end{array}$$

to convert standard minimization form to standard (maximization) form,
the new objective is $-z$, the negative of the original
the new linear constraints are the negatives of the original, $\sum_{j=1}^{n}(-a_{ij})x_j \le -b_i$
nonnegativity remains the same

Remark. we'll see many more "standard forms"!

resource allocation problems often translate immediately into standard maximization form,
with $a_{ij}, b_i, c_j \ge 0$:
n activities
x_j = the level of activity j
m scarse resources
b_i = the amount of resource i that is available
we seek the level of each activity that maximizes total profit

blending problems often translate immediately into standard minimization form,
with $a_{ij}, b_i, c_j \ge 0$:
n raw materials
x_j = the amount of raw material j
m components of the blend
b_i = requirement on the ith component
we seek the amount of each raw material that minimizes total cost

Free variables

a *free variable* has no sign restriction

we can model a free variable x by replacing it by 2 nonnegative variables p & n with $x = p - n$
replace all occurrences of x by $p - n$

the 2 problems are equivalent:
a solution to the original problem gives a solution to the new problem
with the same objective value

conversely, a solution to the new problem gives a solution to the original problem
with the same objective value

A more economical transformation
the above transformation models k free variables $x_i, i = 1, \ldots, k$ by $2k$ new variables

we can model these k free variables by $k + 1$ new variables:
$f_i = p_i - N$

Remark
`LINDO` variables are automatically nonnegative, unless declared `FREE`

Equality constraints

an equality constraint $\sum_{j=1}^{n} a_j x_j = b$ can be modelled by 2 inequality constraints:
$\sum_{j=1}^{n} a_j x_j \le b$
$\sum_{j=1}^{n} a_j x_j \ge b$

this models k equality constraints $\sum_{j=1}^{n} a_{ij} x_j = b_i$, $i = 1, \ldots, k$ by $2k$ new constraints

we can use only $k + 1$ new constraints, by simply adding together all the $\ge$ constraints:
$\sum_{j=1}^{n} a_{ij} x_j \le b_i$, $i = 1, \ldots, k$
$\sum_{i=1}^{k} \sum_{j=1}^{n} a_{ij} x_j \ge \sum_{i=1}^{k} b_i$

(this works since if $<$ held in one of the first k inequalities,
we'd have $<$ in their sum, contradicting the last inequality)

Example. The 2 constraints
$x + y = 10 \quad y + 5z = 20$
are equivalent to
$x + y \le 10 \quad y + 5z \le 20 \quad x + 2y + 5z \ge 30$

Exercise. This optimization problem

minimize $z = x$
subject to $x > 1$

illustrates why strict inequalities are not allowed. Explain.

Exercise. (**Karmarkar Standard Form**) The *Linear Inequalities (LI)* problem is to find a solution to a given system of linear inequalities or declare the system infeasible. Consider the system

$$\sum_{j=1}^n a_{ij}x_j \le b_i \qquad (i = 1, \ldots, m)$$

(i) Show the system is equivalent to a system in this form:

$$\begin{aligned} \sum_{j=1}^n a_{ij}x_j &= b_i \qquad && (i = 1, \ldots, m) \\ x_j &\ge 0 && (j = 1, \ldots, n) \end{aligned}$$

(ii) Show that we can assume a "normalizing" constraint in (i), i.e., any system in form (i) is equivalent to a system in this form:

$$\begin{aligned} \sum_{j=1}^n a_{ij}x_j &= b_i \qquad && (i = 1, \ldots, m) \\ \sum_{j=1}^n x_j &= 1 && \\ x_j &\ge 0 && (j = 1, \ldots, n) \end{aligned}$$

Hint. Let M be an upper bound to each x_j. (The exercise of Handout#25 gives a formula for M, assuming all a_{ij} & b_i are integers. Thus we can assume in (i), $\sum_{j=1}^n x_j \le nM$. Add this constraint and rescale.

(iii) Show a system in form (ii) is equivalent to a system in this form:

$$\begin{aligned} \sum_{j=1}^n a_{ij}x_j &= 0 \qquad && (i = 1, \ldots, m) \\ \sum_{j=1}^n x_j &= 1 && \\ x_j &\ge 0 && (j = 1, \ldots, n) \end{aligned}$$

(iv) Starting with the system of (iii) construct the following LP, which uses another variable s:

$$\text{minimize } z = s$$

$$\begin{aligned} \text{subject to} \qquad \sum_{j=1}^n a_{ij}x_j - \Big(\sum_{j=1}^n a_{ij}\Big)s &= 0 \qquad && (i = 1, \ldots, m) \\ \sum_{j=1}^n x_j + s &= 1 && \\ x_j,\ s &\ge 0 && (j = 1, \ldots, n) \end{aligned}$$

Show (iii) has a solution if and only if (iv) has optimum cost 0. Furthermore (iv) always has nonnegative cost, and setting all variables equal to $1/(n+1)$ gives a feasible solution.

(iv) is standard form for Karmarkar's algorithm. That is, Karmarkar's algorithm has input an LP of the form

$$\text{minimize } z = \sum_{j=1}^n c_j x_j$$

$$\begin{aligned} \text{subject to} \qquad \sum_{j=1}^n a_{ij}x_j &= 0 \qquad && (i = 1, \ldots, m) \\ \sum_{j=1}^n x_j &= 1 && \\ x_j &\ge 0 && (j = 1, \ldots, n) \end{aligned}$$

where any feasible solution has $z \ge 0$ and $x_j = 1/n$, $j = 1, \ldots, n$ is feasible. The algorithm determines whether or not the optimum cost is 0. (The exercise of Handout#18 shows any LP can be placed into Karmarkar Standard Form.)

we can handle many seemingly nonlinear objective functions
by adding a new variable

Minimax Objectives

Example 1. minimize the maximum of 3 variables x, y, z (subject to various other constraints)

LP: add a new variable u & add new constraints:

minimize u new objective

$u \geq x$ new constraints

$u \geq y$

$u \geq z$

this is a correct model:
for any fixed values of x, y, & z the LP sets u to $\max\{x, y, z\}$, in order to minimize the objective

this trick doesn't work to minimize the min of 3 variables – see Exercise below

Example 2. 3 new technologies can manufacture our product with different costs–
$3x + y + 2z + 100$, $4x + y + 2z + 200$, $3x + 3y + z + 60$
but we're not sure which technology we'll be using
to be conservative we minimize the maximum cost of the 3 technologies

LP: add a new variable u & with new constraints:

minimize u new objective

$u \geq 3x + y + 2z + 100$ new constraints

$u \geq 4x + y + 2z + 200$

$u \geq 3x + 3y + z + 60$

Example 2′. In *makespan scheduling* we want to minimize the completion time of the last job.

in general we can solve LPs with a minimax objective function, i.e.,
minimize $z = \max\{\sum_j c_{kj}x_j + d_k : k = 1, \ldots, r\}$
we add variable u & minimize $z = u$ with new constraints
$u \geq \sum_j c_{kj}x_j + d_k$, $k = 1, \ldots, r$
Note: u is a free variable

similarly we can model maximin objective functions
maximize $z = \min\{\sum_j c_{kj}x_j + d_k : k = 1, \ldots, r\}$
add variable u & maximize $z = u$ with new constraints
$u \leq \sum_j c_{kj}x_j + d_k$, $k = 1, \ldots, r$

we can minimize sums of max terms, e.g.,
minimize $\max\{2x + 3y + 1, x + y + 10\} + \max\{x + 7y, 3x + 3y + 3\}$
but not mixed sums, e.g.,
minimize $\max\{2x + 3y + 1, x + y + 10\} + \min\{x + 7y, 3x + 3y + 3\}$

Special Cases

these useful objective functions all have a concavity restriction–
don't try to remember them, just know the general method!

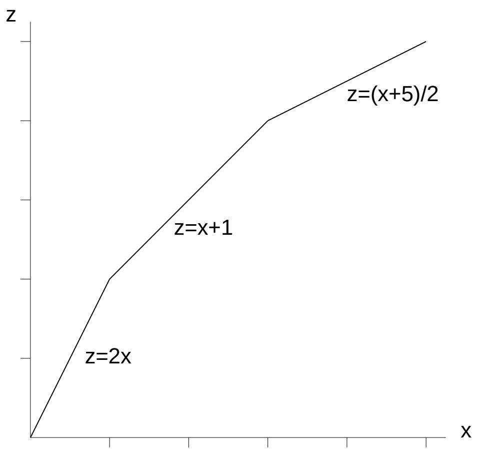
Diminishing Returns (maximizing piecewise linear concave down objective functions)
("concave down" means slope is decreasing)

Example 3.

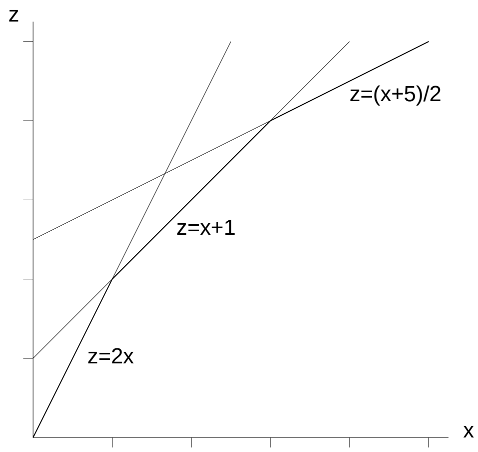
maximizing $z = \begin{cases} 2x & x \le 1 \\ x+1 & 1 \le x \le 3 \\ (x+5)/2 & 3 \le x \le 5 \end{cases}$

is equivalent to

maximizing $\min\{2x, x+1, (x+5)/2\}$



$$z = \begin{cases} 2x & x \le 1 \\ x+1 & 1 \le x \le 3 \\ (x+5)/2 & 3 \le x \le 5 \end{cases}$$



$$\min\{2x, x+1, (x+5)/2\}$$

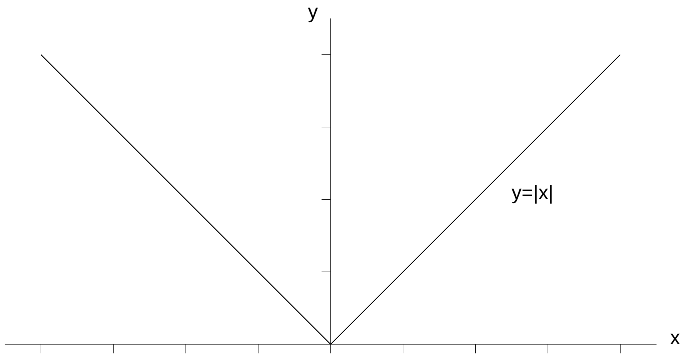
Example 4. maximizing $z = \sum_{j=1}^{n} c_j(x_j)$, where each c_j is a piecewise linear concave down function
the same transformation as Example 3 works

Remark.
Handout #45 gives an alternate solution,
that adds more variables but uses simpler constraints

similarly we can minimize a sum of piecewise linear concave up functions

Absolute Values

Example 5. the objective can contain terms with absolute values, e.g., $|x|$, $3|y|$, $2|x-y-6|$
but the coefficients must be positive in a minimization problem (e.g., $+|x|$)
& negative in a maximization problem (e.g., $-|x|$)



$y=|x|=\max\{x,-x\}$ is concave up.

e.g., minimizing $x+3|y|+2|x-y-6|$
is equivalent to
minimizing $x+3\max\{y,\ -y\}+2\max\{x-y-6,\ -(x-y-6)\}$

Example 6: Data Fitting. (Chvátal, pp.213–223)
we observe data that is known to satisfy a linear relation $\sum_{j=1}^{n} a_{ij}x_j = b_i$, $i=1,\ldots,m$
we want to find the values $x_j, j=1,\ldots,n$ that best approximate the observations

in some cases it's best to use an L_1-approximation
minimize $\sum_{i=1}^{m} \left|b_i - \sum_{j=1}^{n} a_{ij}x_j\right|$ (perhaps subject to $x_j \ge 0$)
and sometimes it's best to use an L_∞-approximation
minimize $\max_i \left|b_i - \sum_{j=1}^{n} a_{ij}x_j\right|$

(when a least-squares, i.e., L_2-approximation, is most appropriate,
we may be able to use calculus, or more generally we use QP – see Handout#42)

Excesses and Shortfalls

Example 7.
in a resource allocation problem, variable x is the number of barrels of high-octane gas produced
the demand for high-octane gas is 10000 barrels
producing more incurs a holding cost of \$25 per barrel
producing less incurs a purchase cost of \$50 per barrel

LP: add new variables e (excess) and s (shortfall)
add terms $-25e - 50s$ to the objective function (maximizing profit)
add constraints $e \geq x - 10000$, $e \geq 0$
& $s \geq 10000 - x$, $s \geq 0$

Remark. we're modelling excess $= \max\{x - 10000, 0\}$, shortfall $= \max\{10000 - x, 0\}$

in general we can model an excess or shortage of a linear function $\sum_j a_j x_j$ & target b
with penalties p_e for excess, p_s for shortage
when we're maximizing or minimizing
if $p_e, p_s \geq 0$

more generally we can allow $p_e + p_s \geq 0$
this allows a reward for excess or shortage (but not both)
to do this add terms $p_e e + p_s s$ to the objective (minimizing cost)
and constraints $\sum_j a_j x_j - b = e - s$, $e \geq 0, s \geq 0$

Exercise. Suppose, similar to Example 1, we want to minimize the minimum of 3 variables x, y, z (subject to various other constraints). Show how to do this by solving 3 LP's, each involving 2 extra constraints.

What is a large LP?

era	m	n	nonzeroes
Dantzig's US economy model	1.5K	4K	40K
2000	10K..100K even 1M	20K..500K even 2M	100K..2M even 6M

the big problems still have only 10..30 nonzeroes per constraint
the bigger problems may take days to solve

Notation: $L =$ (number of bits in the input) (see Handout#69)

Perspective: to understand the bounds, note that Gaussian elimination is time $O(n^3L)$
i.e., $O(n^3)$ operations, each on $O(L)$ bits

Simplex Method

G.B. Dantzig, 1951: "Maximization of a linear function of variables subject to linear inequalities"

visits extreme points, always increasing the profit
can do 2^n pivot steps, each time $O(mn)$

but in practice, simplex is often the method of choice

this is backed up by some theoretic results–
• in a certain model where problems are chosen randomly, average number of pivots is bounded by $\min\{n/2, (m+1)/2, (m+n+1)/8\}$, in agreement with practice
• simplex is polynomial-time if we use "smoothed analysis"– compute average time of a randomly perturbed variant of the given LP

the next 2 algorithms show LP is in $\mathcal{P}$

Ellipsoid Method

L.G. Khachiyan, 1979: "A polynomial algorithm for linear programming"

finds sequence of ellipsoids of decreasing volume, each containing a feasible solution
$O(mn^3L)$ arithmetic operations on numbers of $O(L)$ bits

impractical, even for 15 variables

theoretic tool for developing polynomial time algorithms (Grötschel, Lovasz, Schrijver, 1981)
extends to convex programming, semidefinite programming

Interior Point Algorithm

N. Karmarkar, 1984: "A new polynomial-time algorithm for linear programming"
(*Combinatorica* '84)

navigates through the interior, eventually jumping to optimum extreme point
$O((m^{1.5}n^2 + m^2n)L)$ arithmetic operations on numbers of $O(L)$ bits

in practice, competitive with simplex for large problems

refinements: $O((mn^2 + m^{1.5}n)L)$ operations on numbers of $O(L)$ bits (Vaidya, 1987, and others)

Strong Polynomial Algorithms (for special LPs)

≤ 2 variables per inequality: time $O(mn^3 \log n)$ (Megiddo, 1983)

each $a_{ij} \in \{0, \pm 1\}$ ("combinatorial LPs"): $p(n,m)$ i..e., strong polynomial time (Tardos, 1985)

time $O(m)$ for $n = O(1)$ (Megiddo, 1984)

Randomized Algorithms

relatively simple randomized algorithms achieve average running times that are subexponential
e.g., the only superpolynomial term is $2^{O(\sqrt{n \log n})}$
(Motwani & Raghavan, *Randomized Algorithms*) surveys these

Kelner & Spielman (*STOC 2006*) show a certain randomized version of simplex
runs in polynomial time

Integer Linear Programming

NP-complete

polynomial algorithm for $n = O(1)$, the "basis reduction method" (Lenstra, 1983)

in practice ILPs are solved using a subroutine for LP, and generating additional linear constraints

What is a large ILP?

$m = 100..2\text{K}$ or even 5K; $n = 500..2\text{K}$ or even 5K

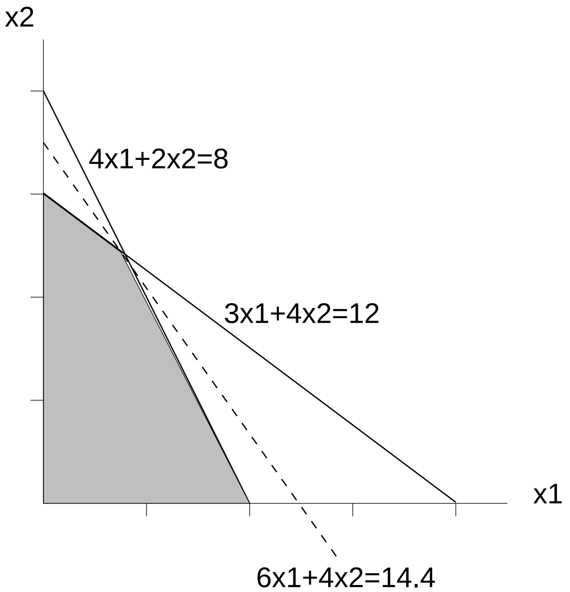
free `LINDO` can handle $m = 300,\ n = 150$ for LP, and $n = 50$ for ILP we solve this LP (a resource allocation problem similar to Handout #1):

maximize $z = 6x_1 + 4x_2$

subject to

$$\begin{aligned} 4x_1 + 2x_2 &\le 8 \\ 3x_1 + 4x_2 &\le 12 \\ x_1, x_2 &\ge 0 \end{aligned}$$

maximum $z = 14.4$ achieved by $(.8, 2.4)$



to start, change inequalities to equations by introducing *slack variables* x_3, x_4

nonnegative, $x_3, x_4 \ge 0$

this gives the *initial dictionary* – it defines the slack variables in terms of the original variables:

$$\begin{array}{rl} x_3 = & 8 - 4x_1 - 2x_2 \\ x_4 = & 12 - 3x_1 - 4x_2 \\ \hline z = & \phantom{12 -{}} 6x_1 + 4x_2 \end{array}$$

this dictionary is associated with the solution $x_1 = x_2 = 0,\ x_3 = 8,\ x_4 = 12,\ z = 0$

the l.h.s. variables x_3, x_4 are the *basic variables*

the r.h.s. variables x_1, x_2 are the *nonbasic variables*

in general a dictionary has the same form as above –

it is a set of equations defining the *basic variables* in terms of the *nonbasic variables*

the solution associated with the dictionary has all nonbasic variables set to zero

the dictionary is *feasible* if this makes all basic variables nonnegative

in which case the solution is a *basic feasible solution* (*bfs*)

increasing x_1 will increase z
we maintain a solution by decreasing x_3 and x_4
$x_3 \ge 0 \Longrightarrow 8 - 4x_1 - 2x_2 \ge 0,\ x_1 \le 2$
$x_4 \ge 0 \Longrightarrow x_1 \le 4$
so set $x_1 = 2$
this gives $z = 12$, and also makes $x_3 = 0$

this procedure is a *pivot step*
we do more pivots to get even better solutions:

the first pivot:
x_1 & x_3 change roles, i.e., x_1 becomes basic, x_3 nonbasic:

1. solve for x_1 in x_3's equation
2. substitute for x_1 in remaining equations

2nd Dictionary

$$\begin{array}{l} x_1 = 2 - \frac{1}{2}x_2 - \frac{1}{4}x_3 \\ x_4 = 6 - \frac{5}{2}x_2 + \frac{3}{4}x_3 \\ \hline z = 12 + x_2 - \frac{3}{2}x_3 \end{array}$$

this dictionary has bfs $x_1 = 2,\ x_4 = 6$
the objective value $z = 12$

the 2nd pivot:
increasing nonbasic variable x_2 will increase z
$\Longrightarrow$ make x_2 the *entering variable*

$x_1 = 2 - \frac{1}{2}x_2 \ge 0 \ \Longrightarrow x_2 \le 4$
$x_4 = 6 - \frac{5}{2}x_2 \ge 0 \ \Longrightarrow x_2 \le 12/5$
$\Longrightarrow$ make x_4 the *leaving variable*

pivot (on entry $\frac{5}{2}$) to get new dictionary

3rd Dictionary

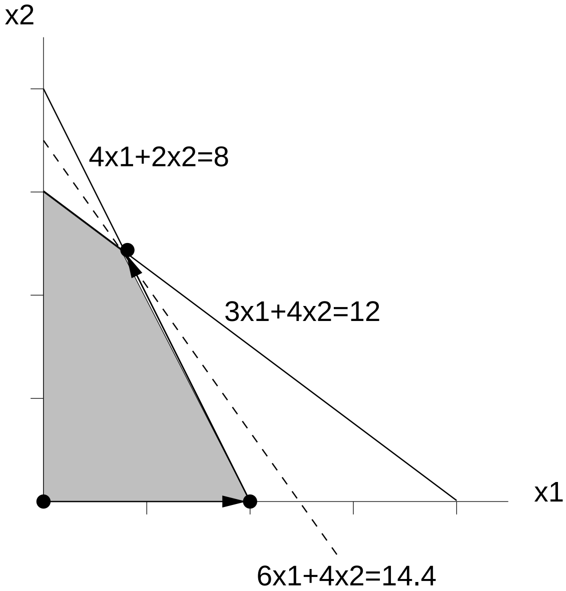
$$\begin{array}{l} x_1 = \frac{4}{5} - \ \frac{2}{5}x_3 + \frac{1}{5}x_4 \\ x_2 = \frac{12}{5} + \frac{3}{10}x_3 - \frac{2}{5}x_4 \\ \hline z = \frac{72}{5} - \ \frac{6}{5}x_3 - \frac{2}{5}x_4 \end{array}$$

this dictionary gives solution $x_1 = 4/5,\ x_2 = 12/5$
an optimum solution

Proof. $x_3, x_4 \ge 0 \Longrightarrow z \le 72/5$ $\square$

Geometric View

the algorithm visits corners of the feasible region, always moving along an edge



Dictionaries
start with an LP (∗) in standard form,

$$(*)\qquad \begin{array}{llll} \text{maximize } z = & \sum_{j=1}^n c_j x_j & & \\ \text{subject to} & \sum_{j=1}^n a_{ij} x_j & \le b_i & (i = 1,\ldots,m) \\ & x_j & \ge 0 & (j = 1,\ldots,n) \end{array}$$

introduce m slack variables $x_{n+1}, \ldots, x_{n+m}$ $\quad (x_{n+i} \ge 0)$

the equations defining the slacks & z give the *initial dictionary* for (∗):

$$\begin{array}{ll} x_{n+i} = b_i - \sum_{j=1}^n a_{ij} x_j & (i = 1,\ldots,m) \\ \hline z = \sum_{j=1}^n c_j x_j & \end{array}$$

> A *dictionary* for LP (∗) is determined by a set of m *basic variables* B.
> The remaining n variables are the *nonbasic variables* N.
> $B, N \subseteq \{1,\ldots,n+m\}$
>
> (i) The dictionary has the form
> $$\begin{array}{ll} x_i = \overline{b}_i - \sum_{j\in N} \overline{a}_{ij} x_j & (i \in B) \\ \hline z = \overline{z} + \sum_{j\in N} \overline{c}_j x_j & \end{array}$$
> (ii) $x_j,\ j = 1,\ldots,m+n$, z is a solution of the dictionary
> $\iff$ it is a solution of the initial dictionary

Remarks.
1. a dictionary is a system of equations
   showing how the nonbasic variables determine the values of the basic variables and the objective
   nonnegativity is not a constraint of the dictionary
2. notice the sign conventions of the dictionary
3. B, the set of basic variables, is a *basis*
4. we'll satisfy condition (ii) by deriving our dictionaries from the initial dictionary
   using equality-preserving transformations

a *feasible dictionary* has each $\overline{b}_i \ge 0$
it gives a *basic feasible solution,* $x_i = \overline{b}_i$ $(i \in B)$, $x_i = 0$ $(i \notin B)$

Examples
1. the initial dictionary of a resource allocation problem is a feasible dictionary
2. the blending problem of Handout #1

$$\begin{array}{lrl}
\text{minimize } z = & 6x+4y & \\
\text{subject to} & 4x+2y & \ge 8 \\
& 3x+4y & \ge 12 \\
& x, y & \ge 0
\end{array}$$

has initial dictionary

$$\begin{array}{l}
s_1 = \ \ -8 + 4x + 2y \\
s_2 = -12 + 3x + 4y \\
\hline
z = \qquad\quad 6x + 4y
\end{array}$$

infeasible!

Lemma ["Nonbasic variables are free."] *Let D be an arbitrary dictionary, with basic (nonbasic) variables B (N).*

(i) Any linear relation always satisfied by the nonbasic variables of D has all its coefficients equal to 0, i.e.,

$$\textstyle\sum_{j\in N} \alpha_j x_j = \beta \text{ for all solutions of } D \implies \beta = 0,\ \alpha_j = 0 \text{ for all } j \in N.$$

(ii) Any linear relation

$$\textstyle\sum_{j\in B\cup N} \alpha_j x_j + \beta = \sum_{j\in B\cup N} \alpha'_j x_j + \beta'$$

always satisfied by the solutions of D has the same coefficients on both sides if all basic coefficients are the same, i.e.,

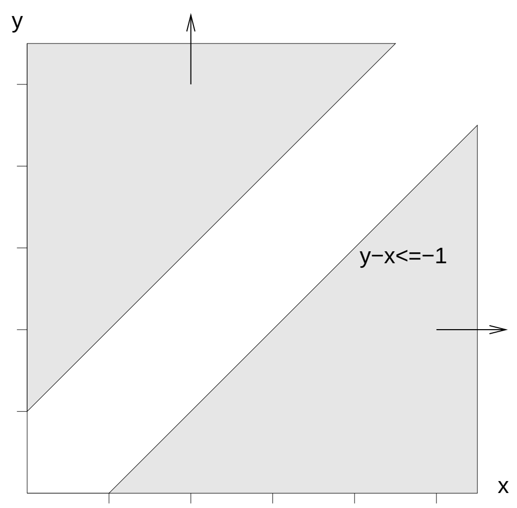
$$\alpha_j = \alpha'_j \text{ for every } j \in B \implies \beta = \beta',\ \alpha_j = \alpha'_j \text{ for every } j \in N. \qquad \square$$

Proof of (i).
setting $x_j = 0$, $j \in N \implies \beta = 0$
setting $x_j = 0$, $j \in N - i$, $x_i = 1 \implies \alpha_i = 0$ $\square$

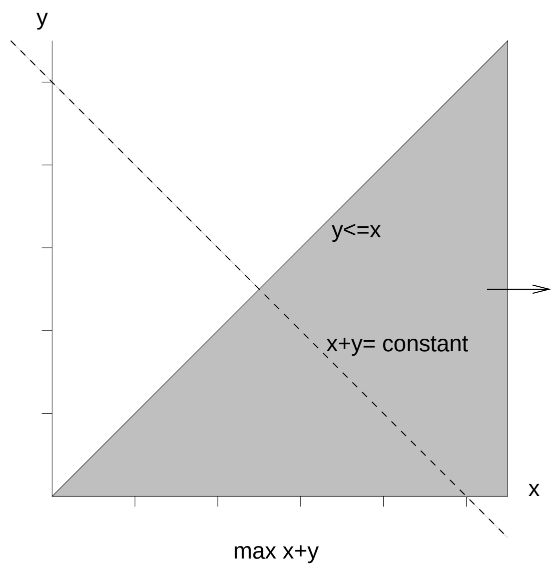
The 3 Possibilities for an LP

any LP either

- (i) has an optimum solution
  it actually has an optimum *bfs* (basic feasible solution)
- (ii) is *infeasible* (no values x_j satisfy all the constraints)
- (iii) is *unbounded* (the objective can be made arbitrarily large)



Infeasible LP



Unbounded LP

we'll show (i) – (iii) are the *only* possibilities for an LP
part of the Fundamental Theorem of Linear Programming

What Constitutes a Solution to an LP?

for (i): an optimum solution – an optimum bfs is even better!

for (ii): a small number ($\leq n+1$) of inconsistent constraints

for (iii): a "line of unboundedness" – on the boundary is best!

in real-world modelling situations, (ii) & (iii) usually indicate errors in the model
the extra information indicates how to fix the model

the basic simplex algorithm is good for conceptualization and hand-calculation
although it ignores 2 issues

Initialization
Construct a feasible dictionary
often, as in a resource allocation problem, initial dictionary is feasible
i.e., all $b_i \geq 0$

Main Loop
Repeat the following 3 steps
until the Entering Variable Step or Leaving Variable Step stops
a_{ij}, b_i, c_j all refer to the current dictionary

Entering Variable Step
If every $c_j \leq 0$, stop, the current basis is optimum
Otherwise choose any (nonbasic) s with $c_s > 0$

Leaving Variable Step
If every $a_{is} \leq 0$, stop, the problem is unbounded
Otherwise choose a (basic) r with $a_{rs} > 0$ that minimizes b_i/a_{is}

Pivot Step
Construct a dictionary for the new basis as follows:
(i) Solve for x_s in the equation for x_r

$$x_s = (b_r/a_{rs}) - \sum_{j \in N'} (a_{rj}/a_{rs}) x_j$$

$N' = N - \{s\} \cup \{r\}$ is the new set of nonbasic variables

note $a_{rr} = 1$ by definition

(ii) Substitute this equation in the rest of the dictionary,
so all right-hand sides are in terms of N' $\square$

in the Entering Variable Step usually > 1 variable has positive cost
the *pivot rule* specifies the choice of entering variable
e.g., the *largest coefficient rule* chooses the entering variable with maximum c_s

the computation in the Leaving Variable Step is called the *minimum ratio test*

Efficiency (Dantzig, *LP & Extensions*, p.160)
in practice the algorithm does between m & $2m$ pivot steps, usually $< 3m/2$

for example see the real-life forestry LP in Chvátal, Ch.11
simplex finds the optimum for 17 constraints in 7 pivots

Deficiencies of the Basic Algorithm

we need to add 2 ingredients to get a complete algorithm:
in general, how do we find an initial feasible dictionary?
how do we guarantee the algorithm halts?

our goal is to show the basic simplex algorithm always halts with the correct answer
assuming we repair the 2 deficiencies of the algorithm
we achieve the goal by proving 6 properties of the algorithm

1. *Each dictionary constructed by the algorithm is valid.*

Proof.
each Pivot Step replaces a system of equations by an equivalent system □

2. *Each dictionary constructed by the algorithm is feasible.*

Proof.
after pivotting, any basic $i \neq s$ has $x_i = b_i - a_{is} \underbrace{(b_r/a_{rs})}_{\text{new } x_s,\ \geq 0}$

$a_{is} \leq 0 \implies x_i \geq b_i \geq 0$
$a_{is} > 0 \implies b_i/a_{is} \geq b_r/a_{rs}$ (minimum ratio test) $\implies b_i \geq a_{is}b_r/a_{rs}$ □

3. *The objective value never decreases:*
It increases in a pivot with $b_r > 0$ and stays the same when $b_r = 0$.

this property shows how the algorithm makes progress toward an optimum solution

Proof.
in the dictionary before the pivot, $z = \overline{z} + \sum_{j\in N} c_j x_j$
the objective value is $\overline{z}$ before the pivot
let it be z' after the pivot

the new bfs has $x_s = b_r/a_{rs}$
thus $z' = \overline{z} + c_s(b_r/a_{rs})$
since $c_s > 0$, $z' \geq \overline{z}$
if $b_r > 0$ then $z' > \overline{z}$ □

4. *Every $c_j \leq 0 \implies$ current basis is optimum.*
"local optimum is global optimum"

Proof.
consider the objective in the current dictionary, $z = \overline{z} + \sum_{j\in N} c_j x_j$
current objective value $= \overline{z}$
any feasible solution has all variables nonnegative $\implies$ its objective value is $\leq \overline{z}$ □

5. *In the Leaving Variable Step, every $a_{is} \leq 0 \implies$ the LP is unbounded.*

Proof.

$$\text{set } x_j = \begin{cases} t & j = s \\ b_j - a_{js}t & j \in B \\ 0 & j \notin B \cup s \end{cases}$$

this is a feasible solution for every $t \geq 0$
its objective value $z = \overline{z} + c_s t$ can be made arbitrarily large $\square$

the simplex algorithm can output this line of unboundedness

Properties 4 – 5 show the algorithm is correct if it stops (i.e., it is *partially correct*)

6. *If the algorithm doesn't stop, it cycles, i.e.,*
   it repeats a fixed sequence of pivots ad infinitum.

Proof.
Claim: there are a finite number of distinct dictionaries
the claim implies Property 6, *assuming* the pivot rule is deterministic

Proof of Claim:
there are $\leq \binom{n+m}{m}$ bases B
each basis B has a unique dictionary
  to show this suppose we have 2 dictionaries for the same basis
  let x_i be a basic variable and consider its equation in both dictionaries,
    $x_i = b_i - \sum_{j \in N} a_{ij} x_j = b'_i - \sum_{j \in N} a'_{ij} x_j$
  nonbasic variables are free $\Longrightarrow$ the equations are the same, i.e., $b_i = b'_i$, $a_{ij} = a'_{ij}$
  similarly, $\overline{z} = \overline{z}'$, $c_j = c'_j$ $\square$

Cycling
in a cycle, the objective z stays constant (Property 3 shows this is necessary for cycling)

so each pivot has $b_r = 0$ (Property 3)

thus the entering variable stays at 0, and the solution $(x_1, \ldots, x_n)$ does not change

Chvátal pp. 31–32 gives an example of cycling (see Handout #48)

Degeneracy
a basis is *degenerate* if one or more basic variables $= 0$
  degeneracy is necessary for cycling
  but simplex can construct a degenerate basis without cycling:
    b_r needn't be 0
    even if $b_r = 0$ we needn't be in a cycle
    although such a pivot does not change the objective value (see Property 3)

(*i*) degeneracy is theoretically unlikely in a random LP, but seems to always occur in practice!

(*ii*) if there is a tie for leaving variable, the new basis is degenerate (see proof of Property 2)

Exercise. Prove the converse: A pivot step gives a nondegenerate dictionary if it starts with a nondegenerate dictionary and has no tie for leaving variable.

Handout #11 adds a rule so we never cycle
in fact, each pivot increases z
this guarantees the algorithm eventually halts with the correct answer

Handout #13 shows how to proceed when the initial dictionary is infeasible

tableaus are an abbreviated representation of dictionaries,
suitable for solving LPs by hand, and used in most LP texts

a *tableau* is a labelled matrix that represents a dictionary,
e.g., here's the initial dictionary of Handout#5 & the corresponding tableau:

$$\begin{array}{rcl} x_3 = & 8 & - 4x_1 - 2x_2 \\ x_4 = & 12 & - 3x_1 - 4x_2 \\ \hline z = & & 6x_1 + 4x_2 \end{array}$$

	x_1	x_2	x_3	x_4	z	b
x_3	4	2	1			8
x_4	3	4		1		12
z	-6	-4			1	0

To get the tableau representing a given dictionary
label the columns with the variable names, followed by z & b
label each row with the corresponding basic variable (from the dictionary),
the last row with z
in each dictionary equation move all variables to l.h.s.
so the equations become $x_i + \sum \overline{a}_{ij} x_j = \overline{b}_i$, $z - \sum \overline{c}_j x_j = \overline{z}$
copy all numeric coefficients (with sign) in the dictionary
into the tableau's corresponding matrix entry

Remarks.

1. a coefficient $\overline{a}_{ij}$ ($\overline{c}_j$) in the dictionary becomes $\overline{a}_{ij}$ ($-\overline{c}_j$) in the tableau
2. Chvátal uses the opposite sign convention in the z row
   `LINDO` uses the same sign convention

To execute the simplex algorithm with tableaus
add a new rightmost column to the tableau, for the ratios in the minimum ratio test
star the pivot element a_{rs} (its row has the minimum ratio)
Pivot Step:
get new pivot row by dividing by the pivot element
relabel the pivot row to x_s (the entering variable)
decrease each row i (excepting the pivot row but including the objective row)
by a_{is} times the (new) pivot row

Solution of the Example LP by Tableaus

Initial Tableau

	x_1	x_2	x_3	x_4	z	b	ratio
x_3	4^*	2	1			8	2
x_4	3	4		1		12	4
z	-6	-4			1	0	

1st Pivot

	x_1	x_2	x_3	x_4	z	b	ratio
x_1	1	.5	.25			2	4
x_4		2.5^*	$-.75$	1		6	2.4
z		-1	1.5		1	12	

Optimum Tableau

	x_1	x_2	x_3	x_4	z	b
x_1	1		.4	$-.2$		.8
x_2		1	$-.3$	.4		2.4
z			1.2	.4	1	14.4

Example 2.

LP: maximize $z = x - y$
subject to

$$\begin{aligned} -x+y &\le 2 \\ ax+y &\le 4 \\ x,y &\le 0 \end{aligned}$$

Initial Tableau

	s_1	s_2	x	y	z	b	ratio
s_1	1	0	-1	1		2	
s_2	0	1	a^*	1		4	$4/a$
z			-1	1	1	0	

this ratio test assumes $a > 0$
if $a \le 0$ the initial tableau has an unbounded pivot
corresponding to the line $y = 0$ (parametrically $x = t, y = 0, s_1 = 2 + t, s_2 = 4 - at$)

1st Pivot (Optimum)
let $\alpha = 1/a$

	s_1	s_2	x	y	z	b
s_1	1	α		$1+\alpha$		$2+4\alpha$
x	0	α	1	α		4α
z		α		$1+\alpha$	1	4α

$\mathbf{R}^n$ – n-dimensional space, i.e., the set of all *vectors* or *points* $(x_1, \ldots, x_n)$

let a_j, $j = 1, \ldots, n$ and b be real numbers with some $a_j \neq 0$
hyperplane – all points of $\mathbf{R}^n$ satisfying $\sum_{j=1}^{n} a_j x_j = b$

e.g.: a line $ax + by = c$ in the plane $\mathbf{R}^2$, a plane $ax + by + cz = d$ in 3-space, etc.
a hyperplane is an $(n-1)$-dimensional space

(*closed*) *half-space* – all points of $\mathbf{R}^n$ satisfying $\sum_{j=1}^{n} a_j x_j \geq b$

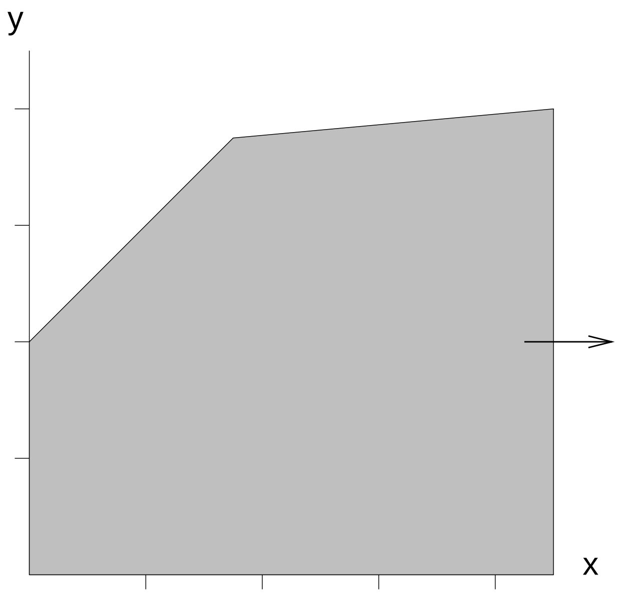
convex polyhedron – an intersection of a finite number of half-spaces
convex polytope – a convex polyhedron that is bounded

let P be a convex polyhedron
the *hyperplanes of* P – the hyperplanes corresponding to the half-spaces of P

this may include "extraneous" hyperplanes that don't change the intersection

P contains various polyhedra –

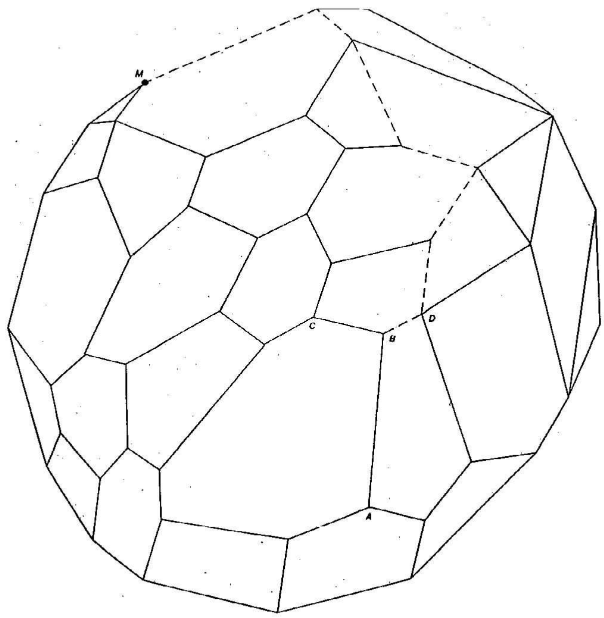
face – P intersected with some of the hyperplanes of P, or $\emptyset$ or P



this unbounded convex polyhedron has
3 vertices, 4 edges (facets), 9 faces total

Special Faces

vertex – 0-dimensional face of P
i.e., a point of P that is the unique intersection of n hyperplanes of P
edge – 1-dimensional face of P
i.e., a line segment that is the intersection of P and $n-1$ hyperplanes of P
can be a ray or a line
facet – $(n-1)$-dimensional face of P



A famous 3D-convex polyhedron

2 vertices of P are *adjacent* if they are joined by an edge of P

a *line* in $\mathbf{R}^n$ has a *parameterized form*, $x_j = m_j t + b_j$, $j = 1, \ldots, n$

Geometry of LPs

the feasible region of an LP is a convex polyhedron P

the set of all optima of an LP form a face
e.g., a vertex, if the optimum is unique
$\emptyset$, if the LP is infeasible or unbounded
an unbounded LP has a *line of unboundedness*, which can always be chosen as an edge
P, if the objective is constant on P

Geometric View of the Simplex Algorithm

(see Handout#47 for proofs of these facts)

consider the problem in activity space (no slacks)

a bfs is a vertex of P

plus n hyperplanes of P that define it

a degenerate bfs is the intersection of $> n$ hyperplanes of P

may (or may not) correspond to $> n$ facets intersecting at a point

(see also Chvátal, pp. 259–260)

corresponds to > 1 dictionary

(nondegeneracy corresponds to "general position" in geometry)

a nondegenerate pivot moves from one vertex, along an edge, to an adjacent vertex

a degenerate pivot stays at the same vertex

the path traversed by the simplex algorithm, from initial vertex to final (optimum) vertex, is the *simplex path*

note the objective function always increases as we move along the simplex path

Hirsch Conjecture. (1957, still open)

any 2 vertices of a convex polyhedron are joined by a simplex path of length $\le m$

actually all the interesting relaxations of Hirsch are also open:

there's a path of length $\le \begin{cases} p(m) \\ p(m,n) \\ p(m,n,L) \quad (L = \text{total } \# \text{ bits in the integers } a_{ij}, b_i) \end{cases}$

here p denotes any polynomial function, and we assume standard form

our geometric intuition can be misleading, e.g.,

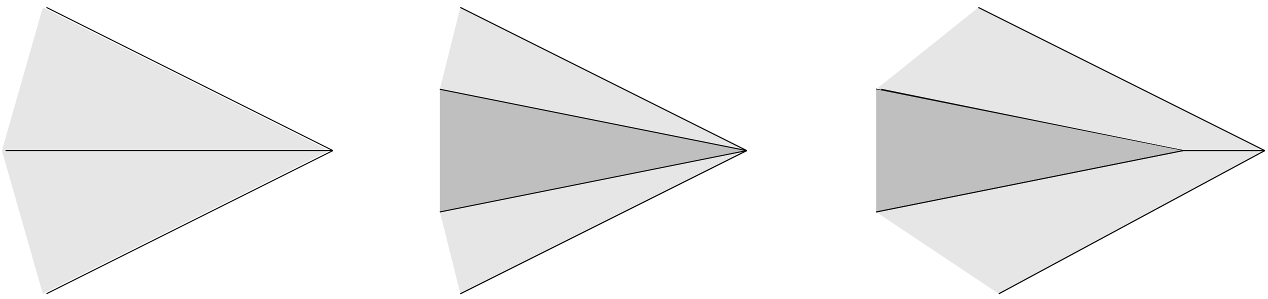
a polyhedron is *neighborly* if every 2 vertices are adjacent

in any dimension ≥ 4 there are neighborly polyhedra with *arbitrarily many* vertices!

we'll give 2 rules, each ensures the simplex algorithm does not cycle
both rules are easy to implement
but many simplex codes ignore the possibility of cycling, since it doesn't occur in practice
avoiding cycling is important theoretically, e.g.,
needed to prove the Fundamental Theorem of LP

Intuition for Lexicographic Method
degeneracy is an "accident", i.e., $> n$ hyperplanes intersect in a common point
a random LP is *totally nondegenerate*, i.e., it has no degenerate dictionary

our approach is to "perturb" the problem, so only n hyperplanes intersect in a common point
$\Longrightarrow$ there are no degenerate bfs's $\Longrightarrow$ the simplex algorithm doesn't cycle



3 planes meet at a vertex | the 3 planes & a 4th meet at a vertex | moving the 4th plane forward gives 2 vertices

The Perturbed LP
given an LP in standard form,

$$\begin{array}{lll} \text{maximize } z = & \sum_{j=1}^n c_j x_j & \\ \text{subject to} & \sum_{j=1}^n a_{ij} x_j \le b_i & (i = 1, \ldots, m) \\ & x_j \ \ge \ 0 & (j = 1, \ldots, n) \end{array}$$

replace each right-hand side b_i by $b_i + \epsilon_i$, where
$0 < \epsilon_m \ll \epsilon_{m-1} \ldots \ll \epsilon_1 \ll 1 = \epsilon_0 \qquad (*)$

Remarks
1. the definition $\epsilon_0 = 1$ comes in handy below

2. it's tempting to use a simpler strategy, replacing b_i by $b_i + \epsilon$
   i.e., use the same perturbation in each inequality
   Chvátal p.34 shows this is incorrect, simplex can still cycle
   the constraints must be perturbed by linearly independent quantities
   we'll see this is crucial in our proof of correctness

3. the appropriate values of ϵ_i are unknown for $i > 0$, and difficult to find
   we finesse this problem by treating the ϵ_i as variables with the above property $(*)$!

imagine executing the simplex method on the perturbed problem
we'll get expressions like $2 + \epsilon_1 - 5\epsilon_2$ as b terms get combined
call such expressions *linear combinations* (of the ϵ_i's)
& simpler expressions that are just numbers, like 2 – call these *scalars*
so a linear combination has the form $\sum_{i=0}^{m} \beta_i \epsilon_i$ where each β_i is a scalar

in any dictionary,
the *coefficients* are a_{ij} & c_j, $i = 1, \ldots, m$, $j = 1, \ldots, n$
the *absolute terms* are b_i, $i = 1, \ldots, m$ & $\overline{z}$

Lemma 1. *Suppose we do a pivot step on a dictionary where*
every coefficient is a scalar, &
every absolute term is a linear combination of the ϵ_i*'s.*
The resulting dictionary has the same form.

Proof. (intuitively the pivots are determined by the a's)
an equation in a dictionary has the form $x_i = \alpha_i - a_{is}x_s - \ldots$
rewriting the pivot equation gives $x_s = (\alpha_r/a_{rs}) - \sum_{j \neq s}(a_{rj}/a_{rs})x_j$
substituting for x_s in each equation other than the pivot equation
preserves every coefficient as a scalar
& every absolute term as a linear combination $\Box$

How Do We Perform the Minimum Ratio Test in the Perturbed Problem?

we want to choose the row i minimizing α_i/a_{is}, a linear combination
so we need to compare linear combinations

$(*)$ tells us we should compare linear combinations using
lexicographic order, i.e., dictionary order, $<_\ell$
e.g., $5\epsilon_0 + 2\epsilon_2 + 9\epsilon_4 \ <_\ell\ 5\epsilon_0 + 3\epsilon_2 - \epsilon_5$
in general for linear combinations $\beta = \sum_{i=0}^{m} \beta_i \epsilon_i$ & $\gamma = \sum_{i=0}^{m} \gamma_i \epsilon_i$,
$\beta <_\ell \gamma \iff$ for some index j, $0 \le j \le m$, $\beta_j < \gamma_j$ and $\beta_i = \gamma_i$ for $i < j$
thus β_0 is most significant, β_1 is next most significant, etc.

this lexical comparison corresponds to an ordinary numeric comparison:
take $\epsilon_i = \delta^i$ with $\delta > 0$ small enough
the above comparison becomes $5 + 2\delta^2 + 9\delta^4 \ < 5 + 3\delta^2 - \delta^5$, i.e., $9\delta^4 + \delta^5 < \delta^2$
it suffice to have $10\delta^4 < \delta^2$, $\delta < 1/\sqrt{10}$

The Lexicographic Method
start with the perturbed problem
execute the simplex algorithm
  choose any pivot rule you wish(!)
  but use lexical order in the minimum ratio test

here's the key fact:

Lemma 2. *The perturbed LP is* <u>*totally nondegenerate*</u>*, i.e.,*
  in any dictionary equation $x_k = \sum_{i=0}^{m} \beta_i \epsilon_i - \sum_{j \in N} \overline{a}_j x_j$,
  the first sum is not lexically 0, i.e., some $\beta_i \neq 0$ *(in fact* $i > 0$*).*

Remark. of course the most significant nonzero β_i will be positive

Proof.
recall the definition of each slack variable in the initial dictionary:
  $x_{n+i} = b_i + \epsilon_i - \sum_{j=1}^{n} a_{ij} x_j$
substitute these equations in the above equation for x_k
this gives an equation involving x_j, $j = 1, \ldots, n$ & ϵ_i, $i = 1, \ldots, m$
  that holds for *any* values of these variables
  so each variable has the same coefficient on both sides of the equation

Case 1. x_k *is a slack variable in the initial dictionary.*
say $k = n + i$, so the l.h.s. has the term ϵ_i
to get ϵ_i on the r.h.s. we need $\beta_i = 1$

Case 2. x_k *is a decision variable in the initial dictionary.*
to get x_k on the r.h.s., some nonbasic slack variable x_{n+i} has $\overline{a}_{n+i} \neq 0$
to cancel the term $-\overline{a}_{n+i}\epsilon_i$ we must have $\beta_i = \overline{a}_{n+i} \neq 0$ $\square$

every pivot in the lexicographic method increases z, lexicographically
  by Lemma 2 & Handout#8, Property 3
so the lexicographic method eventually halts
  with a dictionary giving an optimum or unbounded solution

this dictionary corresponds to a dictionary for the given LP
  take all $\epsilon_i = 0$
& gives an optimum or unbounded solution to the original LP!

Remarks.

1. our original intuition is correct:
   there are numeric values of ϵ_i that give a perturbed problem $\ni$
   the simplex algorithm does exactly the same pivots as the lexicographic method

   just take $\epsilon_i = \delta^i$ with $\delta > 0$ small enough
   this is doable since there are a finite number of pivots

2. many books prove the key Lemma 2 using linear algebra
   (simple properties of inverse matrices)

   Chvátal finesses linear algebra with dictionaries

3. perturbation is a general technique in combinatorial computing
   e.g., any graph has a unique minimum spanning tree if we perturb the weights

4. smoothed analysis (Handout#4) computes the time to solve an LP $\mathcal{L}$
   by averaging over perturbed versions of $\mathcal{L}$
   where we randomly perturb the a_{ij}'s and the b_i's

the choice of entering variable is limited to the *eligible variables*
i.e., those with cost coefficient $c_i > 0$
a pivot rule specifies the entering variable

Common Pivot Rules

Largest Coefficient Rule ("nonbasic gradient", "Dantzig's rule")
choose the variable with maximum c_i; stop if it's negative

this rule depends on the scaling of the variables
e.g., formulating the problem in terms of $x_1' = x_1/10$
makes x_1 10 times more attractive as entering variable

Largest Increase Rule ("best neighbor")
choose the variable whose pivot step increases z the most
(a pivot with x_s entering & x_r leaving increases z by $c_s b_r/a_{rs}$)

in practice this rule decreases the number of iterations but increases the total time

Least Recently Considered Rule
examine the variables in cyclic order, $x_1, x_2, \ldots, x_n, x_1, \ldots$
at each pivot step, start from the last entering variable
the first eligible variable encountered is chosen as the next entering variable

used in many commercial codes

Devex, Steepest Edge Rule ("all variable gradient")
choose variable to make the vector from old bfs to new
as parallel as possible to the cost vector
recent experiments indicate this old rule is actually very efficient
in the dual LP

Open Problem
Is there a pivot rule that makes the simplex algorithm run in polynomial time?

Bland's Rule

nice theoretic properties
slow in practice, although related to the least recently considered rule

Smallest-subscript Rule (Bland, 1977)
if more than one entering variable or leaving variable can be chosen,
always choose the candidate variable with the smallest subscript

Theorem. *The simplex algorithm with the smallest-subscript rule never cycles.*

Proof.
consider a sequence of pivot steps forming a cycle,
i.e., it begins and ends with the same dictionary
we derive a contradiction as follows

let F be the set of all subscripts of variables that enter (and leave) the basis during the cycle
let $t \in F$

let D be a dictionary in the cycle that gives a pivot where x_t leaves the basis
similarly D^* is a dictionary giving a pivot where x_t enters the basis
(note that x_t can enter and leave the basis many times in a cycle)

dictionary D: basis B
coefficients a_{ij}, b_i, c_j
next pivot: x_s enters, x_t leaves

dictionary D^*: coefficients c^*_j
next pivot: x_t enters

Claim: $c_s = c^*_s - \sum_{i\in B} c^*_i a_{is}$

Proof of Claim:
the pivot for D corresponds to solutions $x_s = u,\ x_i = b_i - a_{is}u$, $i \in B$, remaining $x_j = 0$
these solutions satisfy D (although they may not be feasible)

the cost of such a solution varies linearly with u:
dictionary D shows it varies as $c_s u$
dictionary D^* shows it varies as $(c^*_s - \sum_{i\in B} c^*_i a_{is})u$

these two functions must be the same! this gives the claim $\diamondsuit$

we'll derive a contradiction by showing the l.h.s. of the Claim is positive but the r.h.s. is nonpositive

$c_s > 0$: since x_s is entering in D's pivot

to make the r.h.s. nonpositive, choose t as the largest subscript in F

$c^*_s \le 0$: otherwise x_s is nonbasic in D^* & D^*'s pivot makes x_s entering ($s < t$)

$c^*_i a_{is} \ge 0$ for $i \in B$:

Case $i = t$:
$a_{ts} > 0$: since x_t is leaving in D's pivot
$c^*_t > 0$: since x_t is entering in D^*'s pivot

Case $i \in B - F$:
$c^*_i = 0$: since x_i never leaves the basis

Case $i \in B \cap (F - t)$:
$a_{is} \le 0$: since $b_i = 0$ (any variable of F stays at 0 throughout the cycle – see Handout #8) but x_i isn't the leaving variable in D's pivot
$c^*_i \le 0$: otherwise x_i is nonbasic in D^* & D^*'s pivot makes x_i entering ($i < t$) $\square$ (Wow!)

Remarks

1. Bland's discovery resulted from using matroids to study the sign properties of dictionaries

2. *stalling* – when a large number of consecutive pivots stay at the same vertex
   Bland's rule can stall – see Handout#49

3. interesting results have been proved on randomized pivot rules
   e.g., Kalai (*STOC* '92) shows this pivot rule gives subexponential average running time:

   choose a random facet F that passes through the current vertex
   recursively move to an optimum point on F

given a standard form LP—

$$\begin{aligned} &\text{maximize } z = \textstyle\sum_{j=1}^{n} c_j x_j \\ &\text{subject to } \textstyle\sum_{j=1}^{n} a_{ij} x_j \le b_i \qquad (i = 1, \ldots, m) \\ &\qquad\qquad\qquad\quad x_j \ge 0 \qquad (j = 1, \ldots, n) \end{aligned}$$

if all b_i are nonnegative the initial dictionary is feasible,
so the basic simplex algorithm solves the problem
if some $b_i < 0$ we solve the problem as follows:

The Two-Phase Method
Phase 1: find a feasible dictionary (or detect infeasibility)
Phase 2: solve the given LP with the simplex algorithm, starting with the feasible dictionary

we've already described Phase 2; we'll use simplex to do Phase 1 too!

The Phase 1 LP

$$\begin{aligned} &\text{minimize } x_0 \\ &\text{subject to } \textstyle\sum_{j=1}^{n} a_{ij} x_j - x_0 \le b_i \qquad (i = 1, \ldots, m) \\ &\qquad\qquad\qquad\qquad\quad x_j \ge 0 \qquad (j = 0, \ldots, n) \end{aligned}$$

Motivation:
the minimum $= 0 \iff$ the given LP is feasible
but if the given LP is feasible, will we get a feasible dictionary for it?

x_0 is sometimes called an *artificial variable*

before describing Phase 1, here's an example:

the given LP has constraints
$x_1 - x_2 \ge 1, \quad 2x_1 + x_2 \ge 2, \quad 7x_1 - x_2 \le 6 \quad x_1, x_2 \ge 0$

(the first constraint $7x_1 - 7x_2 \ge 7$ is inconsistent with the last $7x_1 - 7x_2 \le 7x_1 - x_2 \le 6$)

put the constraints in standard form:
$-x_1 + x_2 \le -1, \quad -2x_1 - x_2 \le -2, \quad 7x_1 - x_2 \le 6 \quad x_1, x_2 \ge 0$

Phase 1 starting dictionary:	1st pivot: x_0 enters, x_4 leaves
(infeasible)	(achieving Phase 1 feasibility)
$x_3 = -1 + x_1 - x_2 + x_0$	$x_3 = 1 - x_1 - 2x_2 + x_4$
$x_4 = -2 + 2x_1 + x_2 + x_0$	$x_0 = 2 - 2x_1 - x_2 + x_4$
$x_5 = 6 - 7x_1 + x_2 + x_0$	$x_5 = 8 - 9x_1 + x_4$
$w = -x_0$	$w = -2 + 2x_1 + x_2 - x_4$

2nd pivot: x_2 enters, x_3 leaves

$$\begin{array}{l}
x_2 = \frac{1}{2} - \frac{1}{2}x_1 - \frac{1}{2}x_3 + \frac{1}{2}x_4 \\
x_0 = \frac{3}{2} - \frac{3}{2}x_1 + \frac{1}{2}x_3 + \frac{1}{2}x_4 \\
x_5 = 8 \ - 9x_1 \qquad\qquad + x_4 \\
\hline
w \ = -\frac{3}{2} + \frac{3}{2}x_1 - \frac{1}{2}x_3 - \frac{1}{2}x_4
\end{array}$$

last pivot: x_1 enters, x_5 leaves

$$\begin{array}{l}
x_2 = \ldots \\
x_0 = \ldots \\
x_1 = \frac{8}{9} \qquad\qquad + \frac{1}{9}x_4 - \frac{1}{9}x_5 \\
\hline
w \ = -\frac{1}{6} - \frac{1}{2}x_3 - \frac{1}{3}x_4 - \frac{1}{6}x_5
\end{array}$$

optimum dictionary

the optimum w is negative $\implies$ the given problem is infeasible

Exercise 1. Prove that throughout Phase 1, the equation for w and x_0 are negatives of each other.

General Procedure for Phase 1

1. starting dictionary D_0
in the Phase 1 LP, minimizing x_0 amounts to maximizing $-x_0$
introduce slack variables $x_j,\ j = n+1, \ldots, n+m$ to get dictionary D_0 for Phase 1:

$$\begin{array}{ll}
x_{n+i} = b_i - \sum_{j=1}^n a_{ij}x_j + x_0 & \qquad (i = 1, \ldots, m) \\
\hline
w = \qquad\qquad\qquad\quad -x_0 &
\end{array}$$

D_0 is infeasible

2. feasible dictionary
to get a feasible dictionary pivot with x_0 entering, x_{n+k} leaving (we'll choose k momentarily)

$$\begin{array}{ll}
x_0 = -b_k + \sum_{j=1}^n a_{kj}x_j + x_{n+k} & \\
x_{n+i} = b_i - b_k - \sum_{j=1}^n (a_{ij} - a_{kj})x_j + x_{n+k} & \qquad (i = 1, \ldots, m,\ i \neq k) \\
\hline
w = \quad b_k - \sum_{j=1}^n a_{kj}x_j - x_{n+k} &
\end{array}$$

to make this a feasible dictionary choose $b_k = \min\{b_i : 1 \le i \le m\}$
this makes each basic variable nonnegative, since $b_i \ge b_k$

3. execute the simplex algorithm, starting with this feasible dictionary
choose x_0 to leave the basis as soon as it becomes a candidate
then stop ($x_0 = 0 \implies$ optimal Phase 1 solution)

obviously the simplex algorithm halts with an optimum solution
the Phase 1 LP is bounded ($w \le 0$) so it has an optimum

4. Phase 1 ends when the simplex algorithm halts:

Case 1. Phase 1 terminates when x_0 leaves the basis

let D^* be the optimum Phase 1 dictionary, with basis B
(since $x_0 = 0$, D^* gives a feasible solution to given LP)
transform D^* to a feasible dictionary D for the given LP, with basis B:

1. drop all terms involving x_0 from D^*

2. replace the objective function w with an equation for z:
eliminate the basic variables from the given equation for z

$$z = \sum_{j \in B} c_j \underbrace{x_j}_{\substack{\uparrow \\ \text{substitute}}} + \sum_{j \notin B} c_j x_j$$

D is a valid dictionary for the given LP:

Proof.
D^* has the same solutions as D_0
hence D^* & D_0 have same solutions with $x_0 = 0$
i.e., ignoring objective functions,
D has the same solutions as the initial dictionary of the given LP □

now execute Phase 2: run the simplex algorithm, starting with dictionary D

Exercise 1 (cont'd). Prove that in Case 1, the last row of D^* is always $w = -x_0$.

Case 2. Phase 1 terminates with x_0 basic.

In this case the given LP is infeasible

Proof.
it suffices to show that the final (optimum) value of x_0 is > 0
equivalently, no pivot step changes x_0 to 0:

suppose a pivot step changes x_0 from positive to 0
x_0 was basic at the start of the pivot, and could have left the basis
(it had the minimum ratio)
in this case Phase 1 makes x_0 leave the basis □

Remark.
the "big-M" method solves 1 LP instead of 2
it uses objective function $z = \sum_{j=1}^{n} c_j x_j - M x_0$
where M is a symbolic value that is larger than any number encountered

A Surprising Bonus

if an LP is infeasible we'd like our algorithm to output succinct evidence of infeasibility

in our example infeasible LP
the objective of the final dictionary shows how the given constraints imply a contradiction:

using the given constraints in standard form,
add $\frac{1}{2} \times$ (1st constraint) $+ \frac{1}{3} \times$ (2nd constraint) $+ \frac{1}{6} \times$ (3rd constraint), i.e.,

$\frac{1}{2}(-x_1 + x_2 \le -1) + \frac{1}{3}(-2x_1 - x_2 \le -2) + \frac{1}{6}(7x_1 - x_2 \le 6)$
simplifies to $0 \le -\frac{1}{6}$, a contradiction!

relevant definition: a *linear combination of inequalities* is
the sum of multiples of each of the inequalities
the original inequalities must be of the same type ($\le, \ge, <, >$)
the multiples must be nonnegative
we combine the l.h.s.'s & the r.h.s.'s

Phase 1 Infeasibility Proof

in general, suppose Phase 1 halts with optimum objective value $w^* < 0$
consider the last (optimal) dictionary
suppose slack s_i has coefficient $-\overline{c}_i$, $i = 1, \ldots, m$ $\quad (\overline{c}_i \ge 0)$

multiply the ith constraint by $\overline{c}_i$ and add all m constraints
this will give a contradiction, (nonnegative #) $\le w^* < 0$

we show this always works in Handout#32,p.2

LINDO Phase 1 (method sketched in Chvátal, p.129)

Phase 1 does not use any artificial variables. Each dictionary uses a different objective function: The Phase 1 objective for dictionary D is

$$w = \sum x_h$$

where the sum is over all (basic) variables x_h having negative values in D.

Tableau:
Row 1 gives the coefficients, in the current dictionary, of the given objective function. The last row (labelled `ART`) gives the coefficients of the current Phase 1 cost function. This row is constructed by adding together all rows that have negative b_i's (but keeping the entries in the basic columns equal to 0).

Simplex Iteration.
In the following, a_{ij}, b_i and c_j refer to entries in the current `LINDO` tableau (not dictionary!); further, c_j are the Phase 1 cost coefficients, i.e., the entries in the `ART` row. The value of the Phase 1 objective (bottom right tableau entry) is negative.

Entering Variable Step.
If every c_j is ≥ 0 stop, the problem is infeasible.

Otherwise choose a (nonbasic) s with $c_s < 0$.

Leaving Variable Step.
Choose a basic r that minimizes this set of ratios:

$$\{\frac{b_i}{a_{is}} : a_{is} > 0 \text{ and } b_i \geq 0, \text{ or } a_{is} < 0 \text{ and } b_i < 0\}.$$

Pivot Step.
Construct the tableau for the new basis (x_s enters, x_r leaves) except for the `ART` row. If every b_i is nonnegative the current bfs is feasible for the given LP. Proceed to Phase 2.

Otherwise construct the `ART` row by adding the rows of all negative b_i's and zeroing the basic columns.

Exercises.
1. Justify the conclusion of infeasibility in the Entering Variable Step. *Hint.* Show a feasible solution implies an equation (nonnegative number) = (negative number), using $0 \leq \sum x_h < 0$.

2. Explain why the set of ratios in the Leaving Variable Step is nonempty. If it were empty we'd be in trouble!

3. Explain why any variable that is negative in the current dictionary started out negative and remained so in every dictionary.

4. Explain why Phase 1 eventually halts, assuming it doesn't cycle. *Hint.* Show a pivot always increases the current objective function (even when we switch objectives!).

5. Explain why the following is probably a better Leaving Variable Step:
Let $POS = \{i : a_{is} > 0 \text{ and } b_i \geq 0\}$.
Let $NEG = \{i : a_{is} < 0 \text{ and } b_i < 0\}$.
If $POS \neq \emptyset$ then r is the minimizer of $\{\frac{b_i}{a_{is}} : i \in POS\}$.
Otherwise r is the minimizer of $\{\frac{b_i}{a_{is}} : i \in NEG\}$.

recall the standard form LP—

$$
\begin{aligned}
&\text{maximize } z = \textstyle\sum_{j=1}^{n} c_j x_j \\
&\text{subject to } \textstyle\sum_{j=1}^{n} a_{ij} x_j \le b_i \qquad (i = 1, \ldots, m) \\
&\qquad\qquad\qquad\qquad x_j \ge 0 \qquad (j = 1, \ldots, n)
\end{aligned}
$$

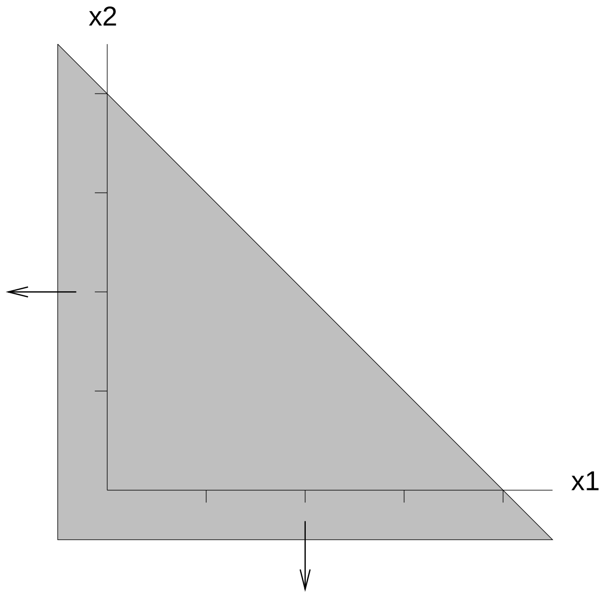
Phase 1 proves the following fact:
A feasible LP in standard form has a basic feasible solution.

geometrically this says the polyhedron of the LP has a corner point

Example 1. this fact needn't hold if the LP is not in standard form, e.g., the 1 constraint LP

$x_1 + x_2 \le 5$

(no nonnegativity constraints) is feasible but has no corner point:



we've now completely proved our main result:

Fundamental Theorem of LP. *Consider any LP in standard form.*

(i) Either the LP has an optimum solution or the objective is unbounded or the constraints are infeasible.

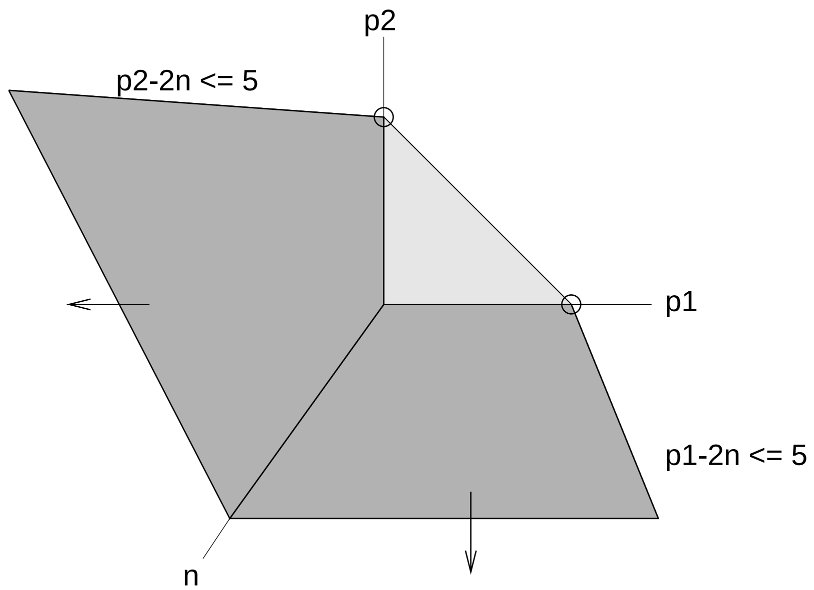
(ii) If the LP is feasible then there is a basic feasible solution.

(iii) If the LP has an optimum solution then there is a basic optimum solution. □

Example 1 cont'd.
adopting the objective function $x_1 + x_2$,
& transforming to standard form by the substitutions $x_j = p_j - n$, gives the LP

$$\begin{array}{ll} \text{maximize } z = p_1 + p_2 - 2n & \\ \text{subject to } \; p_1 + p_2 - 2n & \le \; 5 \\ \qquad\qquad\qquad p_1, p_2, n & \ge \; 0 \end{array}$$

this LP satisfies the Fundamental Theorem, having 2 optimum bfs's/corner points:



The 2 optimum bfs's are circled.

part (i) of the Fundamental Theorem holds for any LP

Question. Can you think of other sorts of linear problems, not quite in standard form and not satisfying the theorem?

an unbounded LP has an edge that's a line of unboundedness
here's a stronger version of this fact:

Extended Fundamental Theorem (see Chvátal, 242–243)
If the LP is unbounded, it has a basic feasible direction with positive cost.

to explain, start with the definition:

consider an arbitrary dictionary
let B be the set of basic variables
let s be a nonbasic variable, with coefficients a_{is}, $i \in B$ in the dictionary
if $a_{is} \le 0$ for each $i \in B$ then the following values w_j, $j = 1, \ldots, n$
form a *basic feasible direction*:

$$w_j = \begin{cases} 1 & j = s \\ -a_{js} & j \in B \\ 0 & j \notin B \cup s \end{cases}$$

(n above denotes the total number of variables, including slacks)

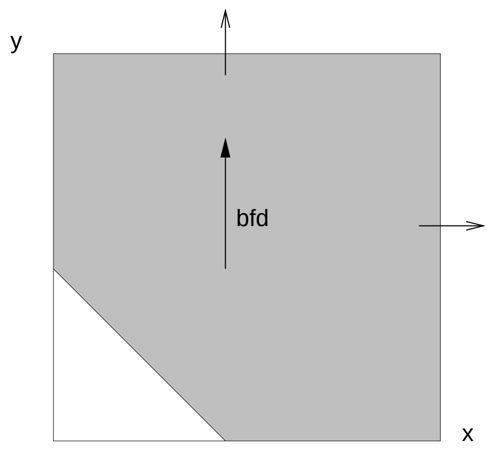
Property of bfd's:
if $(x_1, \ldots, x_n)$ is *any* feasible solution to an LP,
$(w_1, \ldots, w_n)$ is any basic feasible direction, & $t \ge 0$,
then increasing each x_j by tw_j gives another feasible solution to the LP
(prove by examining the dictionary)

Example. take the LP maximize $z = y$
subject to $x+y \ge 1$
$x, y \ge 0$

introducing slack variable s gives feasible dictionary

$$\begin{array}{l} y = 1 - x + s \\ \hline z = 1 - x + s \end{array}$$

basic feasible direction $s = t, y = t, x = 0$



Claim. if an LP is unbounded the simplex algorithm finds a basic feasible direction w_j with $\sum_{j=1}^{n} c_j w_j > 0$ (these c_j's are original cost coefficients)

(n above can be the total number of decision variables)

the Claim implies the Extended Fundamental Theorem

Proof of Claim.
let $\overline{c}_j$ denote the cost cofficients in the final dictionary
& $\overline{z}$ the cost value
let s be the entering variable for the unbounded pivot ($\overline{c}_s > 0$)

in what follows, all sums are over all variables, including slacks

$$\sum_j c_j x_j = \overline{z} + \sum_j \overline{c}_j x_j \qquad \text{for any feasible } x_j$$

$$\sum_j c_j (x_j + w_j) = \overline{z} + \sum_j \overline{c}_j (x_j + w_j) \qquad \text{since } x_j + w_j \text{ is also feasible}$$

subtract to get

$$\sum_j c_j w_j = \sum_j \overline{c}_j w_j = \sum_{j \in B} 0 \cdot w_j + \sum_{j \notin B \cup s} \overline{c}_j \cdot 0 + \overline{c}_s = \overline{c}_s > 0 \qquad \diamondsuit$$

The Dual Problem

consider a standard form LP, sometimes called the *primal problem* –

$$\begin{aligned}
&\text{maximize } z = \textstyle\sum_{j=1}^{n} c_j x_j \\
&\text{subject to } \textstyle\sum_{j=1}^{n} a_{ij} x_j \le b_i \qquad (i = 1, \ldots, m) \\
&\qquad\qquad\qquad\quad x_j \ge 0 \qquad (j = 1, \ldots, n)
\end{aligned}$$

its *dual problem* is this LP –

$$\begin{aligned}
&\text{minimize } z = \textstyle\sum_{i=1}^{m} b_i y_i \\
&\text{subject to } \textstyle\sum_{i=1}^{m} a_{ij} y_i \ge c_j \qquad (j = 1, \ldots, n) \\
&\qquad\qquad\qquad\quad y_i \ge 0 \qquad (i = 1, \ldots, m)
\end{aligned}$$

Caution. this definition only works if the primal is in standard maximization form

Example. find the dual of Chvátal, problem 5.2, p. 69
notice how n & m get interchanged!

Primal Problem

$$\begin{aligned}
\text{maximize} \quad & -x_1 - 2x_2 \\
\text{subject to} \quad & -3x_1 + x_2 \le -1 \\
& x_1 - x_2 \le 1 \\
& -2x_1 + 7x_2 \le 6 \\
& 9x_1 - 4x_2 \le 6 \\
& -5x_1 + 2x_2 \le -3 \\
& 7x_1 - 3x_2 \le 6 \\
& x_1, x_2 \ge 0
\end{aligned}$$

Dual Problem

$$\begin{aligned}
\text{minimize} \quad & -y_1 + y_2 + 6y_3 + 6y_4 - 3y_5 + 6y_6 \\
\text{subject to} \quad & -3y_1 + y_2 - 2y_3 + 9y_4 - 5y_5 + 7y_6 \ge -1 \\
& y_1 - y_2 + 7y_3 - 4y_4 + 2y_5 - 3y_6 \ge -2 \\
& y_1, y_2, y_3, y_4, y_5, y_6 \ge 0
\end{aligned}$$

Exercise. Put the LP
max $-x$ s.t. $x \ge 2$
into standard form to verify that its dual is
min $-2y$ s.t. $-y \ge -1$, $y \ge 0$.
Professor Dull says "Rather than convert to standard form by flipping $x \ge 2$, I'll take the dual first and then flip the inequality. So the dual is
min $2y$ s.t. $-y \le 1$, $y \ge 0$."
Show Dull is wrong by comparing the optimum dual objective values.

Multiplier Interpretation of the Dual, & Weak Duality

the dual LP solves the problem,

Find the best upper bound on the primal objective implied by the primal constraints.

Example cont'd. Prove that the Primal Problem has optimum solution $x_1 = 3/5, x_2 = 0, z = -3/5$.

$z \le (1/3)(-3x_1 + x_2) \Longrightarrow z \le (1/3)(-1) = -1/3$ not good enough

$z \le (1/5)(-5x_1 + 2x_2) \Longrightarrow z \le (1/5)(-3) = -3/5$ yes!

in general we want a linear combination of primal constraints $\ni$

(l.h.s.) $\ge$ (the primal objective)

(r.h.s.) is as small as possible

this corresponds exactly to the definition of the dual problem:

the multipliers are the y_i ($\Longrightarrow$ dual nonnegativity constraints)

dual constraints say coefficient-by-coefficient,

(the linear combination) $\ge$ (the primal objective)

dual objective asks for the best (smallest) upper bound possible

so by definition, (any dual objective value) $\ge$ (any primal objective value)

more formally:

Weak Duality Theorem.
$x_j,\ j = 1, \ldots, n$ *primal feasible* & $y_i,\ i = 1, \ldots, m$ *dual feasible* $\Longrightarrow \sum_{j=1}^{n} c_j x_j \le \sum_{i=1}^{m} b_i y_i$

Proof.

$$\sum_{i=1}^{m} b_i y_i \underset{\substack{\uparrow \\ \text{primal constraints \&} \\ \text{dual nonnegativity}}}{\ge} \sum_{i=1}^{m}\Big(\sum_{j=1}^{n} a_{ij} x_j\Big) y_i \underset{\substack{\uparrow \\ \text{algebra}}}{=} \sum_{j=1}^{n}\Big(\sum_{i=1}^{m} a_{ij} y_i\Big) x_j \underset{\substack{\uparrow \\ \text{dual constraints \&} \\ \text{primal nonnegativity}}}{\ge} \sum_{j=1}^{n} c_j x_j \qquad \square$$

Remarks

1. it's obvious that for a tight upper bound, only tight constraints get combined
   i.e., the multiplier for a loose constraint is 0 – see Handout#19
   it's not obvious how to combine tight constraints to get the good upper bound–
   the dual problem does this

2. How good an upper bound does the dual place on the primal objective?
   it's *perfect*! – see Strong Duality
   it's remarkable that the problem of upper bounding an LP is another LP

3. plot all primal and dual objective values on the x-axis:

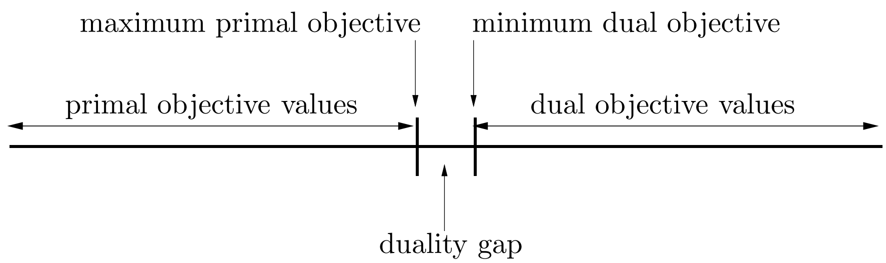


Illustration of weak duality.

   Strong Duality will show the duality gap is actually 0

4. starting with a primal-dual pair of LPs, add arbitrary constraints to each
   Weak Duality still holds (above proof is still valid)
   even though the 2 LPs need no longer form a primal-dual pair

   e.g., we constrain all primal & dual variables to be integers
   this gives a dual pair of integer linear programs
   Weak Duality holds for any dual pair of ILPs
   the duality gap is usually nonzero for dual ILPs

last handout viewed the dual variables as multipliers of primal inequalities
next handout views them as multipliers of dictionary equations in the simplex algorithm
to prepare for this we show the equations of any dictionary
are linear combinations of equations of the initial dictionary

for a given LP, consider the initial dictionary D and any other dictionary $\overline{D}$
D has coefficients a_{ij}, b_i, c_j, basic "slack variables" & nonbasic "decision variables"
$\overline{D}$ has coefficients $\overline{a}_{ij}, \overline{b}_i, \overline{c}_j$
Remark. the same logic applies if D is an arbitrary dictionary

we start by analyzing the objective function:

$\overline{D}$'s cost equation is obtained from D's cost equation, $z = \sum_{j=1}^{n} c_j x_j$,
by adding in, for all i,
$\overline{c}_{n+i}$ times D's equation for x_{n+i}, $x_{n+i} = b_i - \sum_{j=1}^{n} a_{ij} x_j$

in other words:
Lemma 1. *$\overline{D}$'s cost equation, $z = \overline{z} + \sum_{j=1}^{n+m} \overline{c}_j x_j$, is precisely the equation*

$$z = \sum_{j=1}^{n} c_j x_j - \sum_{i=1}^{m} \overline{c}_{n+i}\Big(b_i - \sum_{j=1}^{n} a_{ij} x_j - x_{n+i}\Big).$$

Example. check that the optimum primal dictionary of next handout, p.1 has cost equation
$z = (-x_1 - 2x_2) + 0.2(-3 + 5x_1 - 2x_2 - s_5)$

Proof.
start with two expressions for z:

$$\underbrace{\overline{z} + \sum_{j=1}^{n+m} \overline{c}_j x_j}_{\overline{D}\text{'s equation for } z} \quad = \quad \underbrace{\sum_{j=1}^{n} c_j x_j - \sum_{i=1}^{m} \overline{c}_{n+i}\Big(b_i - \sum_{j=1}^{n} a_{ij} x_j - x_{n+i}\Big)}_{D\text{'s equation for } z,\text{ minus } \sum_{i=1}^{m} \overline{c}_{n+i} \times 0}$$

both sides of the equation are identical, since the slacks have the same coefficient on both sides
(Handout#6, "Nonbasic variables are free") $\square$

Remark. the lemma's equation would be simpler if we wrote $+\overline{c}_{n+i}$ rather than $-\overline{c}_{n+i}$
but the minus sign comes in handy in the next handout

we analyze the equations for the basic variables similarly:
(this is only needed in Handout#20)

$\overline{D}$'s equation for basic variable x_k is $x_k = \overline{b}_k - \sum_{j\in N} \overline{a}_{kj} x_j$
define $\overline{a}_{kj}$ to be 1 if $j = k$ & 0 for any other basic variable
now x_k's equation is a rearrangement of
$0 = \overline{b}_k - \sum_{j=1}^{n+m} \overline{a}_{kj} x_j$

this equation is obtained by adding together, for all i,
$\overline{a}_{k,n+i}$ times D's equation for x_{n+i}, $x_{n+i} = b_i - \sum_{j=1}^{n} a_{ij} x_j$

in other words:
Lemma 2. *$\overline{D}$'s equation for basic variable x_k, $x_k = \overline{b}_k - \sum_{j\in N} \overline{a}_{kj} x_j$, is a rearrangement of the equation*

$$0 = \sum_{i=1}^{m} \overline{a}_{k,n+i}(b_i - \sum_{j=1}^{n} a_{ij} x_j - x_{n+i}).$$

("Rearrangement" simply means move x_k to the l.h.s.)

Examples. Check that in the optimum primal dictionary of next handout, p.1,
the equation for x_1 is a rearrangement of
$0 = -0.2(-3 + 5x_1 - 2x_2 - s_5)$
& in the optimum dual dictionary,
the equation for t_2 is a rearrangement of
$0 = 0.4(1 - 3y_1 + \ldots + 7y_6 - t_1) + (2 + y_1 + \ldots - 3y_6 - t_2)$

Proof.
start with two expressions for 0:

$$\underbrace{\overline{b}_k - \sum_{j=1}^{n+m} \overline{a}_{kj} x_j}_{\overline{D}\text{'s equation for } x_k} \quad = \quad \underbrace{\sum_{i=1}^{m} \overline{a}_{k,n+i}(b_i - \sum_{j=1}^{n} a_{ij} x_j - x_{n+i})}_{D\text{'s equations for the slacks}}$$

again the slacks have the same coefficients on both sides
so both sides are identical $\square$

Moral: it's easy to obtain the equations of a dictionary
as linear combinations of equations of the initial dictionary:
the slack coefficients are the multipliers

Second View of the Dual Variables

the dual LP gives the constraints on the multipliers used by the simplex algorithm
to construct its optimum dictionary

more precisely the simplex algorithm actually solves the dual problem:
in the cost equation of the optimum dictionary
the coefficients $-\overline{c}_{n+i}$, $i = 1, \ldots, m$ form an optimum dual solution
a coefficient $-\overline{c}_j$, $j = 1, \ldots, n$ equals the slack in the jth dual inequality, for this solution

before proving this, here's an example, the primal & dual of Handout#15:
for convenience we denote the slack variables as s_i in the primal, t_j in the dual

Starting Primal Dictionary

$$\begin{aligned} s_1 &= -1 + 3x_1 - x_2 \\ s_2 &= 1 - x_1 + x_2 \\ s_3 &= 6 + 2x_1 - 7x_2 \\ s_4 &= 6 - 9x_1 + 4x_2 \\ s_5 &= -3 + 5x_1 - 2x_2 \\ s_6 &= 6 - 7x_1 + 3x_2 \\ \hline z &= -x_1 - 2x_2 \end{aligned}$$

Starting Dual Dictionary

$$\begin{aligned} t_1 &= 1 - 3y_1 + y_2 - 2y_3 + 9y_4 - 5y_5 + 7y_6 \\ t_2 &= 2 + y_1 - y_2 + 7y_3 - 4y_4 + 2y_5 - 3y_6 \\ \hline z &= y_1 - y_2 - 6y_3 - 6y_4 + 3y_5 - 6y_6 \end{aligned}$$

starting primal dictionary isn't feasible, but this doesn't matter

Optimum Primal Dictionary
(= final Phase 1 dictionary)

$$\begin{aligned} x_1 &= 0.6 + 0.4x_2 + 0.2s_5 \\ s_2 &= 0.4 + 0.60x_2 - 0.2s_5 \\ s_3 &= 7.2 - 6.2x_2 + 0.4s_5 \\ s_4 &= 0.6 + 0.4x_2 - 1.8s_5 \\ s_1 &= 0.8 + 0.2x_2 + 0.6s_5 \\ s_6 &= 1.8 + 0.2x_2 - 1.4s_5 \\ \hline z &= -0.6 - \underbrace{2.4x_2}_{\text{dual slacks}} - \underbrace{0.2s_5}_{\text{dual decisions}} \end{aligned}$$

Optimum Dual Dictionary
(obtained in 1 pivot)

$$\begin{aligned} y_5 &= 0.2 - 0.6y_1 + 0.2y_2 - 0.4y_3 + 1.8y_4 + 1.4y_6 - 0.2t_1 \\ t_2 &= 2.4 - 0.2y_1 - 0.6y_2 + 6.2y_3 - 0.4y_4 - 0.2y_6 - 0.4t_1 \\ \hline z &= 0.6 - \underbrace{0.8y_1 - 0.4y_2 - 7.2y_3 - 0.6y_4 - 1.8y_6}_{\text{primal slacks}} - \underbrace{0.6t_1}_{\text{primal decisions}} \end{aligned}$$

the cost equation of optimum primal dictionary indicates an optimum dual solution
$y_5 = 0.2$, $y_j = 0$ for $j = 1, 2, 3, 4, 6$
& the corresponding slack in the dual constraints
$t_2 = 2.4$, $t_1 = 0$

check out the dual dictionary!

the proof of the next theorem shows the second view is correct in general

our theorem says the primal & dual problems have the same optimum values, as suspected
e.g., the common optimum is -0.6 in our example

Strong Duality Theorem.
If the primal LP has an optimum solution $x_j,\ j = 1, \ldots, n$ *then*

the dual LP has an optimum solution $y_i,\ i = 1, \ldots, m$ *with the same objective value,*

$$\sum_{j=1}^{n} c_j x_j = \sum_{i=1}^{m} b_i y_i.$$

Proof.
consider the optimum primal dictionary found by the simplex method
let bars refer to the optimum dictionary, e.g., $\overline{c}_j$
no bars refer to the given dictionary, e.g., c_j

set $y_i = -\overline{c}_{n+i}$, $i = 1, \ldots, m$

these y_i are the multipliers found by the simplex algorithm, i.e.,
the final cost row is $\sum_{j=1}^{n} c_j x_j + \sum_{i=1}^{m} y_i(b_i - \sum_{j=1}^{n} a_{ij} x_j - x_{n+i})$
note for $j \le n$, $\overline{c}_j = c_j - \sum_{i=1}^{m} a_{ij} y_i = -t_j$, where t_j is the slack in the jth dual constraint

these y_i are dual feasible because the final cost coefficients are nonpositive:
$y_i \ge 0$ since $\overline{c}_{n+i} \le 0$, for $i = 1, \ldots, m$
$\sum_{i=1}^{m} a_{ij} y_i \ge c_j$ since the slack $t_j = -\overline{c}_j \ge 0$, for $j = 1, \ldots, n$

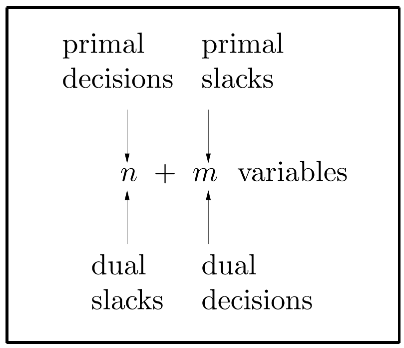
final primal objective value is $\overline{z} = \sum_{i=1}^{m} y_i b_i =$ (the dual objective value)
Weak Duality implies this is the minimum possible dual objective value,
& y_i is optimum $\square$

Remarks.

1. there's no sign flip in `LINDO` tableaux

2. Strong Duality is the "source" of many *minimax theorems* in mathematics
   e.g., the Max Flow Min Cut Theorem (Chvátal, p.370)
   see Handout#63

3. another visualization of primal-dual correspondence:



Variable correspondence in duality

4. many other mathematical programs have duals and strong duality (e.g., see Handout#43)

the dual & primal problems are symmetric, in particular:

Theorem. *The dual of the dual LP is (equivalent to) the primal LP.*
"equivalent to" means they have the same feasible solutions
& the essentially the same objective values

Proof.
the dual in standard form is

$$\begin{aligned} &\text{maximize } \textstyle\sum_{i=1}^{m} -b_i y_i \\ &\text{subject to } \textstyle\sum_{i=1}^{m} -a_{ij} y_i \le -c_j \qquad (j = 1, \ldots, n) \\ &\qquad\qquad\qquad y_i \ge 0 \qquad\qquad (i = 1, \ldots, m) \end{aligned}$$

its dual is equivalent to the primal problem:

$$\begin{aligned} &\text{minimize } \textstyle\sum_{j=1}^{n} -c_j u_j \\ &\text{subject to } \textstyle\sum_{j=1}^{n} -a_{ij} u_j \ge -b_i \qquad (i = 1, \ldots, m) \\ &\qquad\qquad\qquad u_j \ge 0 \qquad\qquad (j = 1, \ldots, n) \qquad \square \end{aligned}$$

for instance in the example of Handout#17
we can read the optimum primal solution from the optimum dual dictionary:
$x_1 = 0.6$, $x_2 = 0$; & also the primal slack values s_i

Remark
to find the dual of an LP with both $\le$ & $\ge$ constraints,
place it into 1 of our 2 standard forms – maximization or minimization
whichever is most convenient

Example 1. Consider the LP

maximize $z = x$ s.t. $x \ge 1$

At first glance it's plausible that the dual is

minimize $z = y$ s.t. $y \le 1, y \ge 0$

To get the correct dual put the primal into standard minimization form,

minimize $z = -x$ s.t. $x \ge 1$, $x \ge 0$

and get the correct dual

maximize $z = y$ s.t. $y \le -1$, $y \ge 0$

Alternatively convert to standard maximization form

maximize $z = x$ s.t. $-x \le -1$, $x \ge 0$

and get dual

minimize $z = -y$ s.t. $-y \ge 1$, $y \ge 0$.

The 3 Possibilities for an LP and its Dual

if an LP has an optimum, so does its dual (Strong Duality)
if an LP is unbounded, its dual is infeasible (Weak Duality)
e.g., in Example 1 the primal is unbounded and the dual is infeasible

these observations & the previous theorem show there are 3 possibilities for an LP and its dual:

(i) both problems have an optimum & optimum objective values are =
(ii) 1 problem is unbounded, the other is infeasible
(iii) both problems are infeasible

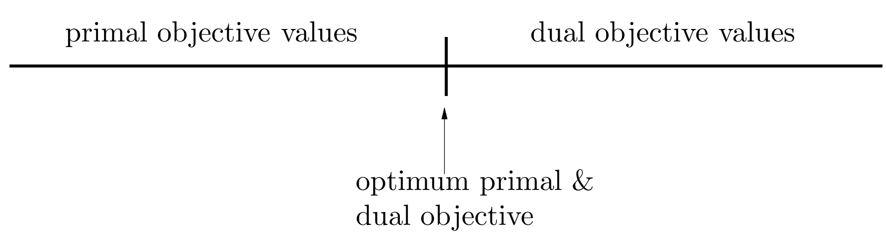
Example of (iii):

$$\begin{array}{lrrl} \text{maximize} & 5x_1 & +\ 6x_2 & \\ \text{subject to} & & 29x_2 & \le -5 \\ & -29x_1 & & \le -6 \\ & & x_1, x_2 & \ge 0 \end{array}$$

this LP is infeasible
the LP is *self-dual*, i.e., it is its own dual
so the dual is infeasible

Exercise. Using matrix notation of Unit 4, show the LP
max $\mathbf{cx}$ s.t. $\mathbf{Ax} \le \mathbf{b},\ \mathbf{x} \ge \mathbf{0}$
is self-dual if $\mathbf{A}$ is skew symmetric and $\mathbf{c} = -\mathbf{b}^{\mathrm{T}}$.

in case (i), plotting all primal and dual objective values on the x-axis gives



in case (ii) the optimum line moves to $+\infty$ or $-\infty$

Exercise. As in the exercise of Handout#2 the *Linear Inequalities (LI)* problem is to find a solution to a given system of linear inequalities or declare the system infeasible. We will show that LI is equivalent to LP, i.e., an algorithm for one problem can be used to solve the other.

(i) Show an LP algorithm can solve an LI problem.

(ii) Show an LI algorithm can solve an LP problem. To do this start with a standard form LP,

$$\begin{array}{lrll}
\text{maximize } z = & \sum_{j=1}^{n} c_j x_j & & \\
\text{subject to} & \sum_{j=1}^{n} a_{ij} x_j & \le b_i & (i = 1, \ldots, m) \\
& x_j & \ge 0 & (j = 1, \ldots, n)
\end{array}$$

and consider the LI problem,

$$\begin{array}{rll}
\sum_{j=1}^{n} a_{ij} x_j & \le b_i & (i = 1, \ldots, m) \\
x_j & \ge 0 & (j = 1, \ldots, n) \\
\sum_{i=1}^{m} a_{ij} y_i & \ge c_j & (j = 1, \ldots, n) \\
y_i & \ge 0 & (i = 1, \ldots, m) \\
\sum_{j=1}^{n} c_j x_j - \sum_{i=1}^{m} b_i y_i & \ge 0 &
\end{array}$$

(ii) together with the exercise of Handout#2 shows any LP can be placed into the standard form required by Karmarkar's algorithm.

rephrase the termination condition of the simplex algorithm in terms of duality:
for any $j = 1, \ldots, n$, $x_j > 0 \implies x_j$ basic $\implies \overline{c}_j = 0 \implies t_j = 0$, i.e,
$\quad x_j > 0 \implies$ the jth dual constraint holds with equality
for any $i = 1, \ldots, m$, $y_i > 0 \implies \overline{c}_{n+i} < 0 \implies x_{n+i}$ nonbasic, i.e.,
$\quad y_i > 0 \implies$ the ith primal constraint holds with equality

here's a more general version of this fact:
as usual assume a standard form primal LP

Complementary Slackness Theorem.
Let $x_j,\ j = 1, \ldots, n$ be primal feasible, $y_i,\ i = 1, \ldots, m$ dual feasible.

x_j is primal optimal & y_i is dual optimal $\iff$
for $j = 1, \ldots, n$, either $x_j = 0$ or $\sum_{i=1}^m a_{ij} y_i = c_j$
and
for $i = 1, \ldots, m$, either $y_i = 0$ or $\sum_{j=1}^n a_{ij} x_j = b_i$.

here's an equivalent formulation:

Complementary Slackness Theorem.
Let $x_j,\ j = 1, \ldots, n$ be primal feasible, $y_i,\ i = 1, \ldots, m$ dual feasible.
Let $s_i,\ i = 1, \ldots, m$ be the slack in the ith primal inequality,
& $t_j,\ j = 1, \ldots, n$ the slack in the jth dual inequality.

x_j is primal optimal & y_i is dual optimal $\iff$
for $j = 1, \ldots, n$, $x_j t_j = 0$ & for $i = 1, \ldots, m$, $y_i s_i = 0$.

Remark CS expresses a fact that's obvious from the multiplier interpretation of duality –
the dual solution only uses tight primal constraints

Proof.
Weak Duality holds for x_j and y_i
let's repeat the proof:

$$\sum_{i=1}^m b_i y_i \ge \sum_{i=1}^m \left(\sum_{j=1}^n a_{ij} x_j\right) y_i = \sum_{j=1}^n \left(\sum_{i=1}^m a_{ij} y_i\right) x_j \ge \sum_{j=1}^n c_j x_j$$

x_j and y_i are both optimal $\iff$ in this proof, the two $\ge$'s are $=$ (Strong Duality)
the first $\ge$ is an $=$ $\iff$ for each $i = 1, \ldots, m$, s_i or y_i is 0
the 2nd $\ge$ is an $=$ $\iff$ for each $j = 1, \ldots, n$, t_j or x_j is 0 $\square$

Remarks.

1. a common *error* is to assume $x_j = 0 \Longrightarrow \sum_{i=1}^m a_{ij} y_i \neq c_j$, or vice versa

2. the simplex algorithm maintains primal feasibility and complementary slackness (previous page) & halts when dual feasibility is achieved

3. Complementary Slackness is the basis of *primal-dual algorithms* (Ch.23)
   they solve LPs by explicitly working on both the primal & dual

   e.g., the Hungarian algorithm for the assignment problem; minimum cost flow problems & primal-dual approximation algorithms for NP-hard problems

Exercise. Show the set of all optimum solutions of an LP is a face.

Testing optimality

complementary slackness gives a test for optimality, of *any* LP solution

given a standard form LP $\mathcal{L}$ –

$$
\begin{aligned}
&\text{maximize } z = \textstyle\sum_{j=1}^n c_j x_j \\
&\text{subject to } \textstyle\sum_{j=1}^n a_{ij} x_j \le b_i \qquad (i = 1, \ldots, m) \\
&\qquad\qquad\qquad\qquad x_j \ge 0 \qquad (j = 1, \ldots, n)
\end{aligned}
$$

let $x_j,\ j = 1, \ldots, n$ be a feasible solution to $\mathcal{L}$
our result says x_j is an optimal solution $\Longleftrightarrow$ it has optimal simplex multipliers:

Theorem. *x_j is optimal $\Longleftrightarrow \exists\ y_i,\ i = 1, \ldots, m \quad \ni$*

$$x_j > 0 \Longrightarrow \sum_{i=1}^m a_{ij} y_i = c_j \tag{1}$$

$$\sum_{j=1}^n a_{ij} x_j < b_i \Longrightarrow y_i = 0 \tag{2}$$

$$\sum_{i=1}^m a_{ij} y_i \ge c_j \qquad (j = 1, \ldots, n) \tag{3}$$

$$y_i \ge 0 \qquad (i = 1, \ldots, m) \tag{4}$$

remembering the optimum cost equation of a dictionary (Handout#17),
$\overline{c}_j = -t_j = c_j - \sum_{i=1}^m a_{ij} y_i$ for $j \le n$, $\quad \overline{c}_{n+i} = -y_i$ for $i \le m$
(3)–(4) say "any variable has nonpositive cost"
(1)–(2) say "basic variables have cost 0"

Proof.
$\Longrightarrow$: Strong Duality shows the optimum y_i exists
Complementary Slackness gives (1) – (2)

$\Longleftarrow$: Complementary Slackness guarantees x_j is optimal $\square$

Application

to check a given feasible solution x_j is optimal
use (2) to deduce the y_i's that vanish
use (1) to find the remaining y_i's (*assuming* a unique solution)
then check (3) – (4)

Examples:
1. Chvátal pp. 64–65
2. check $x_1 = .6, x_2 = 0$ is optimum to the primal of Handout#15, p.1 ($y_5 = .2$, $y_i = 0$ for $i \neq 5$)

the above uniqueness assumption is "reasonable" –
for a nondegenerate bfs x_j, (1)–(2) form a system of m equations in m unknowns
more precisely if k decision variables are nonzero and $m - k$ slacks are nonzero
(1) becomes a system of k equations in k unknowns

satisfying the uniqueness condition:

Theorem. $x_j,\ j = 1, \ldots, n$ *a nondegenerate bfs* $\Longrightarrow$ *system (1)–(2) has a unique solution.*

Proof.
let D be a dictionary for x_j
(1)–(2) are the equations satisfied by the m multipliers for the cost equation of D
so we need only show D is unique
since y_i appears in the cost equation, distinct multipliers give distinct cost equations

uniqueness follows since
x_j corresponds to a unique basis (nondegeneracy) & any basis has a unique dictionary $\square$

Corollary. *An LP with an optimum nondegnerate dictionary has a unique optimum dual solution.*

although the primal can still have many optima –
primal: max $2x_1 + 4x_2$ s.t. $x_1 + 2x_2 \leq 1$, $x_1, x_2 \geq 0$
optimum nondegenerate dictionary: $\begin{array}{c} x_1 = 1 - s - 2x_2 \\ \hline z = 2 - 2s \end{array}$
dual: min y s.t. $y \geq 2, 2y \geq 4, y \geq 0$

Exercise. If the complementary slackness conditions "almost hold", we're "close to" optimality. This principle is the basis for many ILP approximation algorithms. This exercise proves the principle, as follows.

Consider a standard form LP

$$\begin{aligned}
&\text{maximize } z = \textstyle\sum_{j=1}^{n} c_j x_j \\
&\text{subject to } \textstyle\sum_{j=1}^{n} a_{ij} x_j \le b_i \qquad (i = 1, \ldots, m) \\
&\qquad\qquad\qquad\qquad x_j \ge 0 \qquad (j = 1, \ldots, n)
\end{aligned}$$

with optimum objective value z^*. Let x_j, $j = 1, \ldots, n$ be primal feasible & y_i, $i = 1, \ldots, m$ dual feasible, such that these weakened versions of Complementary Slackness hold, for two constants $\alpha, \beta \ge 1$:

$$\begin{aligned}
&\text{for } j = 1, \ldots, n,\ x_j > 0 \text{ implies } \textstyle\sum_{i=1}^{m} a_{ij} y_i \le \alpha c_j; \\
&\text{for } i = 1, \ldots, m,\ y_i > 0 \text{ implies } \textstyle\sum_{j=1}^{n} a_{ij} x_j \ge b_i/\beta.
\end{aligned}$$

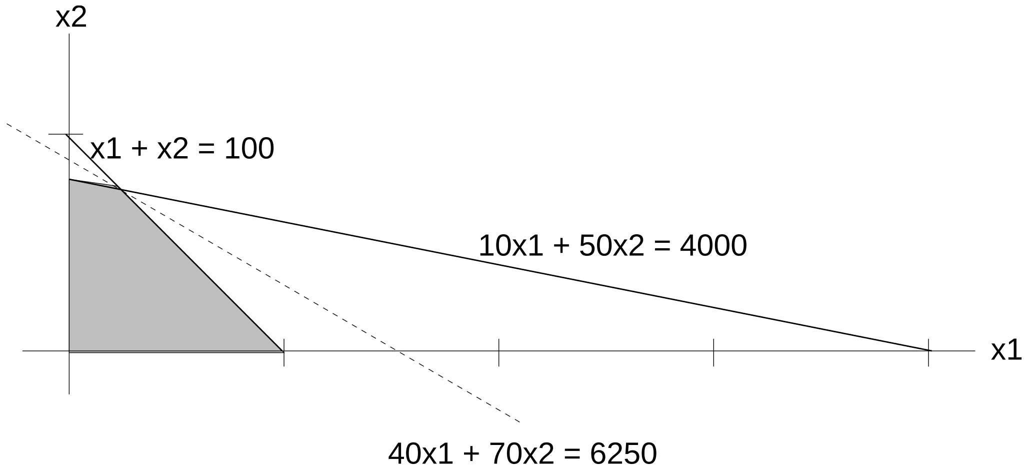
Show that the values x_j, $j = 1, \ldots, n$ solve the given LP to within a factor $\alpha\beta$ of opimality, i.e.,

$$z^* \le \alpha\beta \textstyle\sum_{j=1}^{n} c_j x_j.$$

Hint. Mimic the proof of Weak Duality.

dual variables are prices of resources: y_i is the marginal value of resource i
i.e., y_i is the per unit value of resource i,
assuming just a small change in the amount of resource i

Chvátal's Forestry Example (pp. 67–68)



$x_1 = \#$ acres to fell and regenerate; $x_2 = \#$ acres to fell and plant pine

standard form LP–

$s_1 = 100 - x_1 - x_2$ acreage constraint, 100 acres available

$s_2 = 4000 - 10x_1 - 50x_2$ cash constraint, \$4000 on hand

$z = 40x_1 + 70x_2$ net profit

z gives the net profit from 2 the forestry activities, executed at levels x_1 and x_2

optimum dictionary D^*–

$$x_1 = 25 - 1.25s_1 + .025s_2$$
$$x_2 = 75 + .25s_1 - .025s_2$$
$$z = 6250 - 32.5s_1 - .75s_2$$

suppose (as in Chvátal) there are t more units of resource #2, cash
(t is positive or negative)
in 2nd constraint, $4000 \to 4000 + t$
How does this change dictionary D^*?

since the optimum dual solution is $y_1 = 32.5$, $y_2 = .75$,
Lemma 1 of Handout#16 shows D^*'s objective equation is
(original equation for z) + $32.5\times$ (1st constraint) $+.75\times$ (2nd constraint)
$\Longrightarrow$ the objective equation becomes $z = 6250 + .75t - 32.5s_1 - .75s_2$

D^* is an optimum dictionary as long as it's feasible–
get the constraints of D^* using Lemma 2 of Handout#16

constraint for x_1 is (a rearrangement of) $1.25\times$ (1st constraint) $-.025\times$ (2nd constraint)
$\Longrightarrow x_1 = 25 - .025t - 1.25s_1 + .025s_2$

constraint for x_2 is (a rearrangement of) $-.25\times$ (1st constraint) $+.025\times$ (2nd constraint)
$\Longrightarrow x_2 = 75 + .025t + .25s_1 - .025s_2$

so D^* is optimum precisely when $25 \ge .025t \ge -75$, i.e., $-3000 \le t \le 1000$

in this range, t units of resource #2 increases net profit by $.75t$
i.e., the marginal value of 1 unit of resource #2 is .75, i.e., y_2

thus it's profitable to purchase extra units of resource 2
    at a price of $\le$ (current price) $+0.75$

    i.e., borrow \$1 if we pay back $\le$ \$1.75
    invest \$1 (of our \$4000) in another activity if it returns $\ge$ \$1.75

Remark
packages differ in their sign conventions for dual prices–
    `LINDO` dual price = amount objective <u>improves</u> when r.h.s. increases by 1
    `AMPL` dual price (`.rc`, dual variable) = amount objective <u>increases</u> when increase by 1

Marginal Value Theorem

the dual variables are "marginal values", "shadow prices"
specifically we prove y_i is the marginal value of resource i:

suppose a standard form LP $\mathcal{L}$ has a nondegenerate optimum bfs
 let D^* (with starred coefficients, e.g., z^*) be the optimum dictionary

let $y_i,\ i = 1, \ldots, m$ be the optimum dual solution
 (unique by Handout #19)

for values $t_i,\ i = 1, \ldots, m$, define the "perturbed" LP $\mathcal{L}(t)$:

$$\begin{array}{llr} \text{maximize} & \sum_{j=1}^n c_j x_j & \\ \text{subject to} & \sum_{j=1}^n a_{ij} x_j \le b_i + t_i & (i = 1, \ldots, m) \\ & x_j \ge 0 & (j = 1, \ldots, n) \end{array}$$

Theorem. $\exists\ \epsilon > 0\ \ni$ *for any* $t_i,\ |t_i| \le \epsilon,\ i = 1, \ldots, m,$
 $\mathcal{L}(t)$ *has an optimum & its optimum value is* $z^* + \sum_{i=1}^m y_i t_i$.

Mnemonic. $z = \sum b_i y_i$, so $\partial z / \partial b_i = y_i$

this theorem is stronger than the above example
 the resource amounts can vary independently

Proof.
recall Lemmas 1–2, Handout#16:
in the optimum, nondegenerate dictionary D^* for $\mathcal{L}$,
each equation is a linear combination of equations of the initial dictionary
 cost equation has associated multipliers y_i
 kth constraint equation has associated multipliers u_{ki}

using these multipliers on $\mathcal{L}(t)$ gives dictionary with
 basic values $b_k^* + \sum_{i=1}^m u_{ki} t_i$, cost $z^* + \sum_{i=1}^m y_i t_i$

this solution is optimum as long as it's feasible, i.e., $b_k^* + \sum_{i=1}^m u_{ki} t_i \ge 0$

let $b = \min\{b_k^* : 1 \le k \le m\}$ ($b > 0$ by nondegeneracy)
 $U = \max\{|u_{ki}| : 1 \le k, i \le m\}$ ($U > 0$, else no constraints)
 $\epsilon = b/(2mU)$ ($\epsilon > 0$)

taking $|t_i| \le \epsilon$ for all i makes l.h.s. $\ge b - mU\epsilon = b/2 > 0$ $\square$

More Applications to Economics

Economic Interpretation of Complementary Slackness

$\sum_{j=1}^{n} a_{ij}x_j < b_i \Longrightarrow y_i = 0$
   not all of resource i is used $\Longrightarrow$ its price is 0
   i.e., more of i doesn't increase profit

$\sum_{i=1}^{m} a_{ij}y_i > c_j \Longrightarrow x_j = 0$
   if the resources consumed by activity j are worth more than its (net) profit,
   we won't produce j

Nonincreasing Returns to Scale

we show the value of each resource is nonincreasing, by Weak Duality:

let $\mathcal{L}$ have optimum value z^* & (any) dual optimal solution $y_i,\ i = 1, \ldots, m$
   (y_i is unique of $\mathcal{L}$ is nondegenerate, but we don't assume that)

Theorem. *For any* $t_i,\ i = 1, \ldots, m$ *and any fs* $x_j,\ j = 1, \ldots, n$ *to* $\mathcal{L}(t)$,
   $\sum_{j=1}^{n} c_j x_j \le z^* + \sum_{i=1}^{m} y_i t_i$

Proof.
we repeat the Weak Duality argument:

$$\sum_{j=1}^{n} c_j x_j \le \sum_{j=1}^{n} (\sum_{i=1}^{m} a_{ij}y_i)x_j = \sum_{i=1}^{m} (\sum_{j=1}^{n} a_{ij}x_j)y_i \le \sum_{i=1}^{m} (b_i + t_i)y_i = z^* + \sum_{i=1}^{m} t_i y_i$$

(last step uses Strong Duality) $\square$

in standard maximization form
each $\le$ constraint gives a nonnegative dual variable
each nonnegative variable gives a $\ge$ constraint

we can extend this correspondence to allow equations and free variables
in standard maximization form:
(i) each = constraint gives a free dual variable
(ii) each free variable gives an = constraint
(in all other respects we form the dual as usual)

(i) & (ii) hold in standard minimization form too

Example.

Primal

$$\begin{array}{rr} \text{maximize} & x_1 + 2x_2 + 3x_3 + 4x_4 \\ \text{subject to} & -3x_1 + x_2 + x_3 - x_4 \le -1 \\ & x_1 - x_2 - x_3 + 2x_4 = 1 \\ & -2x_1 + 7x_2 + x_3 - 4x_4 = 6 \\ & 9x_1 - 4x_2 - x_3 + 6x_4 \le 6 \\ & x_1, x_2 \ge 0 \end{array}$$

Dual

$$\begin{array}{rr} \text{minimize} & -y_1 + y_2 + 6y_3 + 6y_4 \\ \text{subject to} & -3y_1 + y_2 - 2y_3 + 9y_4 \ge 1 \\ & y_1 - y_2 + 7y_3 - 4y_4 \ge 2 \\ & y_1 - y_2 + y_3 - y_4 = 3 \\ & -y_1 + 2y_2 - 4y_3 + 6y_4 = 4 \\ & y_1, y_4 \ge 0 \end{array}$$

note that to form a dual, we must still start with a "consistent" primal
e.g., a maximization problem with no $\ge$ constraints

Proof of (i) – (ii).
consider a problem $\mathcal{P}$ in standard form plus additional equations & free variables
we transform $\mathcal{P}$ to standard form & take the dual $\mathcal{D}$
we show $\mathcal{D}$ is equivalent to
the problem produced by using rules (i) – (ii) on $\mathcal{P}$

(i) consider an = constraint in $\mathcal{P}$, $\sum_{j=1}^n a_{ij}x_j = b_i$
it gets transformed to standard form constraints,

$$\sum_{j=1}^n a_{ij}x_j \le b_i$$
$$\sum_{j=1}^n -a_{ij}x_j \le -b_i$$

these constraints give 2 nonnegative variables in $\mathcal{D}$, p_i & n_i
the jth constraint of $\mathcal{D}$ has the terms $a_{ij}p_i - a_{ij}n_i$ & the objective $b_ip_i - b_in_i$
equivalently, $a_{ij}(p_i - n_i)$ & $b_i(p_i - n_i)$
substituting $y_i = p_i - n_i$ gives a free dual variable y_i with terms $a_{ij}y_i$ & b_iy_i

(ii) is similar
in fact just read the above proof backwards $\square$

Exercises.
1. Repeat the proof using our slicker transformations, i.e., just 1 negative variable/1 $\ge$ constraint.

2. Show the dual of an LP in standard maximization form remains the same when we think of all variables as free and $x_j \ge 0$ as a linear constraint.

3. Show dropping a constraint that doesnt change the feasible region doesn't change the dual.

Remarks
1. Equivalent LPs have equivalent duals, as the exercises show.

2. Often we take the primal problem to be, max $\sum c_j x_j$ s.t. $\sum a_{ij}x_j \le b_i$
(optimizing over a general polyhedron)
with dual min $\sum y_i b_i$ s.t. $\sum y_i a_{ij} = c_j,\ y_i \ge 0$

obviously *Weak Duality* & *Strong Duality* still hold for a primal-dual pair

Chvátal proves all this sticking to the interpretation of the dual LP
as an upper bound on the primal

Complementary Slackness still holds–
there's no complementary slackness condition for an equality constraint or a free variable!
(since it's automatic)

Examples. we give 2 examples of LPs with no Complementary Slackness conditions:

1. here's a primal-dual pair where every feasible primal or dual solution is optimum:

$$\begin{array}{ll} \text{maximize } x_1 + 2x_2 & \text{minimize } y_1 \\ \text{subject to } x_1 + 2x_2 = 1 & \text{subject to } \ y_1 = 1 \\ & \qquad\qquad\quad 2y_1 = 2 \end{array}$$

2. in this primal-dual pair, the dual problem is infeasible

$$\begin{array}{ll} \text{maximize } x_1 & \text{minimize } y_1 \\ \text{subject to } x_1 + x_2 = 1 & \text{subject to } y_1 = 1 \\ & \qquad\qquad\quad y_1 = 0 \end{array}$$

Saddle Points in Matrices

Fig.1. Matrices with row minima underlined by leftward arrow
& column maxima marked with upward arrow.

Fact. *In any matrix, (the minimum entry of any row)* $\le$ *(the maximum entry of any column).*

an entry is a *saddle point* if it's the minimum value in its row and the maximum value in its column

a matrix with no duplicate entries has ≤ 1 saddle point (by the Fact)

Example. Fig.1(a) has a saddle point but (b) does not

0-Sum Games & Nash Equilibria

a *finite 2-person 0-sum game* is specified by an $m \times n$ *payoff matrix* a_{ij}
when the `ROW` player chooses i and the `COLUMN` player chooses j,
`ROW` wins a_{ij}, `COLUMN` wins $-a_{ij}$

Example. Fig.1(b) is for the game *Matching Pennies* –
`ROW` & `COLUMN` each choose heads or tails
`ROW` wins \$1 when the choices match & loses \$1 when they mismatch

for Fig.1(a),
`ROW` maximizes her worst-case earnings by choosing a maximin strategy,
i.e., she choose the row whose minimum entry is maximum, row 2
`COLUMN` maximizes her worst-case earnings by choosing a minimax strategy,
i.e., she choose the column whose maximum entry is minimum, column 1

this game is *stable* – in repeated plays,
neither player is tempted to change strategy
this is because entry -1 is a saddle point

in general,
if a payoff matrix has all entries distinct & contains a saddle point,
both players choose it, the game is stable &
(`ROW`'s worst-case winnings) = (`COLUMN`'s worst-case losses) $(*)$
& in repeated plays both players earn/lose this worst-case amount

Remark.
for 0-sum games, stability is the same as a "Nash point":
in any game, a *Nash equilibrium point* is a set of strategies for the players
where no player can improve by unilaterally changing strategies

Example 2. Matching Pennies is *unstable*:

$$\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$$

Fig.2 `ROW` plays row 1 and `COLUMN` plays column 1.
Then `COLUMN` switches to 2. Then `ROW` switches to 2.
Then `COLUMN` switches to 1. Then `ROW` switches to 1.
The players are in a loop!

Example 3. this game is stable, in spite of the embedded cycle from Matching Pennies:

$$\begin{bmatrix} -1 & 2 & -2 \\ -1 & -2 & 2 \\ 0 & 1 & 1 \end{bmatrix}$$

any game with no saddle point is unstable -
there's no Nash equilibrium point, so some player will always switch
`ROW` prefers $\uparrow$ and will switch to it
`COLUMN` prefers $\leftarrow$ and will switch to it
$\therefore$ no saddle point $\Longrightarrow$ 1 or both players always switch

but suppose we allow (more realistic) stochastic strategies –
each player plays randomly, choosing moves according to fixed probabilities

Example.
in Matching Pennies, each player chooses heads or tails with probability 1/2
this is a Nash equilibrium point:
each player has expected winnings = 0
she cannot improve this by (unilaterally) switching strategies –
any other strategy still has expected winnings = 0
the game is stable and $(*)$ holds

but what if `ROW` plays row 1 with probability 3/4?
`COLUMN` will switch to playing column 2 always,
increasing expected winnings from 0 to $1/2 = (3/4)(1) + (1/4)(-1)$
then they start looping as in Fig.2

we'll show that in general, stochastic strategies recover $(*)$

The Minimax Theorem.
For any payoff matrix, there are stochastic strategies for `ROW` & `COLUMN` $\ni$
(`ROW`*'s worst-case expected winnings) = (*`COLUMN`*'s worst-case expected losses).*

Proof.
`ROW` plays row i with probability x_i, $i = 1, \ldots, m$

this strategy gives worst-case expected winnings $\geq z$ iff
`ROW`'s expected winnings are $\geq z$ for each column

to maximize z, `ROW` computes x_i as the solution to the LP

$$\begin{aligned}
&\text{maximize } z \\
&\text{subject to } z - \textstyle\sum_{i=1}^{m} a_{ij} x_i \leq 0 \qquad j = 1, \ldots, n \\
&\qquad\qquad \textstyle\sum_{i=1}^{m} x_i = 1 \\
&\qquad\qquad\qquad x_i \geq 0 \qquad i = 1, \ldots, m \\
&\qquad\qquad\qquad z \text{ unrestricted}
\end{aligned}$$

`COLUMN` plays column j with probability y_j, $j = 1, \ldots, n$

this strategy gives worst-case expected losses $\leq w$ iff
the expected losses are $\leq w$ for each row

to minimize w, `COLUMN` computes y_j as the solution to the LP

$$\begin{aligned}
&\text{minimize } w \\
&\text{subject to } w - \textstyle\sum_{j=1}^{n} a_{ij} y_j \geq 0 \qquad i = 1, \ldots, m \\
&\qquad\qquad \textstyle\sum_{j=1}^{n} y_j = 1 \\
&\qquad\qquad\qquad y_j \geq 0 \qquad j = 1, \ldots, n \\
&\qquad\qquad\qquad w \text{ unrestricted}
\end{aligned}$$

these 2 LPs are duals
both are feasible (set one $x_i = 1$, all others 0)
$\Longrightarrow$ both have the same optimum objective value $\square$

the common optimum is the *value* of the game
obviously if both players use their optimum strategy,
they both earn/lose this worst-case amount

Exercise. Using complementary slackness, check that `ROW` & `COLUMN` have the equal expected winnings.

Example for Minimax Theorem.

$$\begin{array}{cc} & \begin{array}{ccc} & 5/11 & 6/11 \end{array} \\ \begin{array}{c} 7/11 \\ 4/11 \\ \end{array} & \begin{bmatrix} 0 & -2 & 2 \\ 5 & 4 & -3 \\ 2 & 3 & -4 \end{bmatrix} \end{array}$$

Fig.4 Optimum stochastic strategies are given along the matrix borders.
The loop for deterministic play is also shown.

the value of this game is 2/11:

ROW's expected winnings equal

$(7/11)\underbrace{[(5/11)(-2)+(6/11)(2)]}_{2/11}+(4/11)\underbrace{[(5/11)(4)+(6/11)(-3)]}_{2/11}=2/11$

$$\begin{array}{cc} & \begin{array}{cccc} p & p & 1/2-p & 1/2-p \end{array} \\ \begin{array}{c} q \\ q \\ 1/2-q \\ 1/2-q \end{array} & \begin{bmatrix} 0 & 0 & 1 & -1 \\ 0 & 0 & -1 & 1 \\ 1 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 \end{bmatrix} \end{array}$$

Fig.5 Game with 2 copies of Matching Pennies.
General form of optimum strategies is shown, for $0 \le p, q \le 1/2$.

Remarks

1. as shown in Handout#3,
   LP can model a minimax objective function (as shown in `COLUMN`'s LP)
   or a maximin (`ROW`'s LP)

2. in more abstract terms we've shown the following:
   a matrix with a saddle point satisfies $\max_i \min_j \{a_{ij}\} = \min_j \max_i \{a_{ij}\}$
   the Minimax Theorem says *any* matrix has
   a stochastic row vector $\mathbf{x}^*$ & a stochastic column vector $\mathbf{y}^*$ with
   $\min_{\mathbf{y}} \mathbf{x}^* \mathbf{A} \mathbf{y} = \max_{\mathbf{x}} \mathbf{x} \mathbf{A} \mathbf{y}^*$

the matrix representation of LPs uses standard row/column conventions
e.g., here's an example LP we'll call $\mathcal{E}$:

$$\text{maximize } z = \begin{bmatrix} 3 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$\text{subject to } \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \le \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \ge \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Standard Form LP:	*Standard Minimization Form:* (dual)
maximize $\mathbf{cx}$	minimize $\mathbf{yb}$
subject to $\mathbf{Ax} \le \mathbf{b}$	subject to $\mathbf{yA} \ge \mathbf{c}$
$\mathbf{x} \ge \mathbf{0}$	$\mathbf{y} \ge \mathbf{0}$

Conventions
$\mathbf{A}$: $m \times n$ coefficient matrix
$\mathbf{b}$: column vector of r.h.s. coefficients (length m)
$\mathbf{c}$: row vector of costs (length n)
$\mathbf{x}$: column vector of primal variables (length n)
$\mathbf{y}$: row vector of dual variables (length m)

Initial Dictionary
let $\mathbf{x}_S$ be the column vector of slacks
$\mathbf{x}_S = \mathbf{b} - \mathbf{Ax}$

$z = \mathbf{cx}$

more generally:
extend $\mathbf{x}$, $\mathbf{c}$, $\mathbf{A}$ to take slack variables into account
now $\mathbf{x}$ & $\mathbf{c}$ are length $n+m$ vectors; $\mathbf{A} = [\mathbf{A}_0\ \mathbf{I}\ ]$ is $m \times (n+m)$

the standard form LP becomes *Standard Equality Form*:

maximize $\mathbf{cx}$
subject to $\mathbf{Ax} = \mathbf{b}$
$\mathbf{x} \ge \mathbf{0}$

we assume this form has been constructed from standard form, i.e., slacks exist
alternatively we assume $\mathbf{A}$ has rank m, i.e., it contains a basis (see Handout#31)

e.g., our example LP $\mathcal{E}$ has constraints $\begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$

a basis is denoted by B, an ordered list of indices of basic variables
(each index is between 1 & $n+m$)
e.g., in $\mathcal{E}$ the basis of slacks is $3, 4$
when B denotes a basis, N denotes the indices of nonbasic variables
i.e., the complementary subset of $\{1, \ldots, n+m\}$, in any order

Theorem. *B is a basis $\iff$ $\mathbf{A}_B$ is a nonsingular matrix.*

Proof.
$\Longrightarrow$:
the dictionary for B is equivalent to the given system $\mathbf{Ax} = \mathbf{b}$
it shows the bfs for B is the unique solution when we set $\mathbf{x}_N = \mathbf{0}$,
 i.e., the unique solution to $\mathbf{A}_B\mathbf{x}_B = \mathbf{b}$
this implies $\mathbf{A}_B$ is nonsingular (see Handout #55)

$\Longleftarrow$:
let $\mathbf{B}$ denote $\mathbf{A}_B$ (standard notation)

the given constraints are $\mathbf{B}\mathbf{x}_B + \mathbf{A}_N\mathbf{x}_N = \mathbf{b}$

solve for $\mathbf{x}_B$ by multiplying by $\mathbf{B}^{-1}$
 the ith variable of B is the l.h.s. of ith dictionary equation
express objective $z = \mathbf{c}_B\mathbf{x}_B + \mathbf{c}_N\mathbf{x}_N$ in terms of $\mathbf{x}_N$ by substituting for $\mathbf{x}_B$

thus the *dictionary for a basis B* is

$$\begin{array}{l} \mathbf{x}_B = \mathbf{B}^{-1}\mathbf{b} - \mathbf{B}^{-1}\mathbf{A}_N\mathbf{x}_N \\ \hline z = \mathbf{c}_B\mathbf{B}^{-1}\mathbf{b} + (\mathbf{c}_N - \mathbf{c}_B\mathbf{B}^{-1}\mathbf{A}_N)\mathbf{x}_N \end{array} \qquad \square$$

Remark.
the expression $\mathbf{c}_B\mathbf{B}^{-1}$ in the cost equation will be denoted $\mathbf{y}$ (the vector of dual values)
 the cost row corresponds to Lemma 1 of Handout#16–
 looking at the original dictionary, $\mathbf{y}$ is the vector of multipliers

Example.
in $\mathcal{E}$, $B = (1, 2)$ gives $\mathbf{B} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$, $\mathbf{B}^{-1} = \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix}$

dictionary:

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ 2 \end{bmatrix} - \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix}\begin{bmatrix} x_3 \\ x_4 \end{bmatrix}$$

$$z = \begin{bmatrix} 3 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \end{bmatrix} + \left(\begin{bmatrix} 0 & 0 \end{bmatrix} - \begin{bmatrix} 3 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix}\right)\begin{bmatrix} x_3 \\ x_4 \end{bmatrix} = 4 - 2x_3 - x_4$$

$$\mathbf{c}_B\mathbf{B}^{-1} = \begin{bmatrix} 3 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 1 \end{bmatrix}$$

in scalar form,

$$\begin{array}{l} x_1 = 1 - x_3 \\ x_2 = 1 + x_3 - x_4 \\ \hline z = 4 - 2x_3 - x_4 \end{array}$$

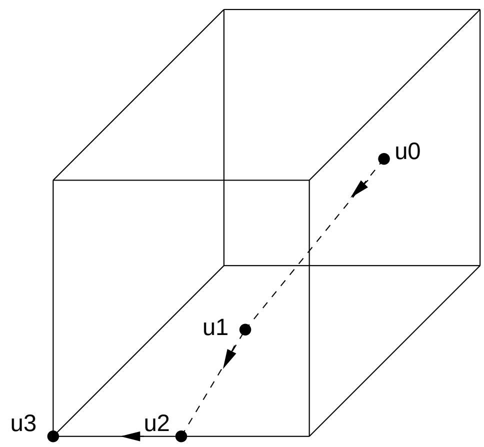
Exercise. (**Rounding Algorithm**) Consider an LP

minimize $\mathbf{cx}$
subject to $\mathbf{Ax} \le \mathbf{b}$

where $\mathbf{A}$ is an $m \times n$ matrix of rank n. (This means $\mathbf{A}$ has n linearly independent columns. Any LP in standard form satisfies this hypothesis.) Let $\mathbf{u}$ be feasible. We will prove the feasible region has a vertex with cost $\le \mathbf{cu}$, unless it has a line of unboundedness.

Let $I \subseteq \{1, \ldots, m\}$ be a maximal set of linearly independent constraints that are tight at $\mathbf{u}$. If $|I| = n$ then $\mathbf{u}$ is a vertex and we're done. So suppose $|I| < n$.

We will find either a line of unboundedness or a new $\mathbf{u}$, of no greater cost, that has a larger set I. Repeating this procedure $\le n$ times gives the desired line of unboundedness or the desired vertex $\mathbf{u}$.



Path $u_0, \ldots, u_3$ taken by rounding algorithm.
Objective = height.

Choose a nonzero vector $\mathbf{w}$ such that $\mathbf{A}_{i\cdot}\mathbf{w} = 0$ for every constraint $i \in I$.

(i) Explain why such a $\mathbf{w}$ exists.

Assume $\mathbf{cw} \le 0$ (if not, replace $\mathbf{w}$ by its negative).

(ii) Explain why every constraint i with $\mathbf{A}_{i\cdot}\mathbf{w} \le 0$ is satisfied by $\mathbf{u}+t\mathbf{w}$ for every $t \ge 0$. Furthermore the constraint is tight (for every $t \ge 0$) if $\mathbf{A}_{i\cdot}\mathbf{w} = 0$.

Let J be the set of constraints where $\mathbf{A}_{i\cdot}\mathbf{w} > 0$.

(iii) Suppose $J = \emptyset$ and $\mathbf{cw} < 0$. Explain why $u + t\mathbf{w}$, $t \ge 0$ is a line of unboundedness.

(iv) Suppose $J \ne \emptyset$. Give a formula for τ, the largest nonnegative value of t where $\mathbf{u} + t\mathbf{w}$ is feasible. Explain why $u + \tau\mathbf{w}$ has a larger set I, and cost no greater than $\mathbf{u}$.

(v) The remaining case is $J = \emptyset$ and $\mathbf{cw} = 0$. Explain why choosing the vector $-\mathbf{w}$ gets us into the previous case.

This proof is actually an algorithm that converts a given feasible point $\mathbf{u}$ into a vertex of no greater cost (or a line of unboundedness). The algorithm is used in Karmarkar's algorithm.

(vi) Explain why any polyhedron $\mathbf{Ax} \le \mathbf{b}$ where $\mathbf{A}$ has rank n has a vertex.

we've described the *standard simplex algorithm* –
it works with completely specified dictionaries/tableaus
the *revised simplex algorithm* implements the standard simplex more efficiently
using linear algebra techniques

to understand the approach recall the dictionary for a basis B:

$$\begin{array}{l} \mathbf{x}_B = \mathbf{B}^{-1}\mathbf{b} - \mathbf{B}^{-1}\mathbf{A}_N\mathbf{x}_N \\ \hline z = \mathbf{c}_B\mathbf{B}^{-1}\mathbf{b} + (\mathbf{c}_N - \mathbf{c}_B\mathbf{B}^{-1}\mathbf{A}_N)\mathbf{x}_N \end{array}$$

1. most of $\mathbf{B}^{-1}\mathbf{A}_N\mathbf{x}_N$ isn't needed –
we only use the column of the entering variable
so the revised simplex algorithm doesn't compute it!

2. this could be done by computing/maintaining $\mathbf{B}^{-1}$
but inverting a matrix is slow, inaccurate, & most importantly we may lose sparsity
e.g., Chvátal p.96, ex. 6.12
instead we'll use routines that solve linear systems $\mathbf{Ax} = \mathbf{b}$, $\mathbf{yA} = \mathbf{c}$

Data Structures for Revised Simplex Algorithm

A, b, c all refer to the given LP, which is in Standard Equality Form

the *basis heading* B is an ordered list of the m basic variables
$\mathbf{B}$ denotes $\mathbf{A}_B$, i.e., the m basic columns of $\mathbf{A}$ (ordered according to B)
$\mathbf{x}^*_B$ is the vector of current basic values, $\mathbf{B}^{-1}\mathbf{b}$ (ordered according to B)

to find the entering variable we need the current dictionary's cost equation
we'll compute the vector $\mathbf{y} = \mathbf{c}_B\mathbf{B}^{-1}$
notice its appearance twice in the cost equation
at termination $\mathbf{y}$ is the optimum dual vector (the simplex multipliers)

to find the leaving variable we need the entering variable's coefficients in the current dictionary
this is the vector $\mathbf{d} = \mathbf{B}^{-1}\mathbf{A}_{.s}$

Revised Simplex Algorithm, High-level

Entering Variable Step
Solve $\mathbf{yB} = \mathbf{c}_B$
Choose any (nonbasic) $s \ni c_s > \mathbf{yA}_{.s}$
If none exists, stop, B is an optimum basis

Leaving Variable Step
Solve $\mathbf{Bd} = \mathbf{A}_{.s}$
Let t be the largest value $\ni \mathbf{x}^*_B - t\mathbf{d} \geq \mathbf{0}$
If $t = \infty$, stop, the problem is unbounded
Otherwise choose a (basic) r whose component of $\mathbf{x}^*_B - t\mathbf{d}$ is zero

Pivot Step
In basis heading B replace r by s (this redefines $\mathbf{B}$)
$\mathbf{x}^*_B \leftarrow \mathbf{x}^*_B - t\mathbf{d}$
In $\mathbf{x}^*_B$, replace entry for r (now 0) by t $\Box$

Correctness & Efficiency

Note: in Standard Equality Form, $n \ge m$ (i.e., # variables $\ge$ # equations)

Entering Variable Step
a nonbasic variable x_j has current cost coefficient $c_j - \mathbf{y}\mathbf{A}_{\cdot j}$

to save computation it's convenient to take the entering variable as
the first nonbasic variable with positive cost

time for this step: $O(m^3)$ to solve the system of equations
plus $O(m)$ per nonbasic variable considered, $O(mn)$ in the worst case

Leaving Variable Step
$x_s = t$, $\mathbf{x}_B = \mathbf{x}^*_B - t\mathbf{d}$, & all other variables 0 satisfies the dictionary equations
since $\mathbf{x}^*_B$ does
and increasing x_s to t decreases the r.h.s. by $\mathbf{d}t$
so $x_s = t$ is chosen to preserve nonnegativity

if $\mathbf{x}^*_B - t\mathbf{d} \ge \mathbf{0}$ for all $t \ge 0$, it gives a line of unboundedness, i.e.,
as t increases without bound, so does z
(since x_s has positive cost coefficient)

time: $O(m^3)$

Pivot Step
time: $O(m)$

our worst-case time estimates show revised simplex takes time $O(m^3 + mn)$ per iteration
not an improvement: standard simplex takes time $O(mn)$ per iteration (Pivot Step)

but we'll implement revised simplex to solve each system of equations
taking advantage of the previous solution
we'll take advantage of *sparsity* of the given LP
real-life problems are sparse
this is the *key* to the efficiency of the revised simplex algorithm

Gaussian Elimination

solves a linear system $\sum_{j=1}^{n} a_{ij}x_j = b_i,\ i = 1, \ldots, n$ in 2 steps:

1. rewrite the equations so each variable has a "substitution equation" of the form $x_i = b_i + \sum_{j=i+1}^{n} a_{ij}x_j$, as follows:

   for each variable x_j, $j = 1, \ldots, n$
   - choose a remaining equation with $a_{ij} \neq 0$
   - rewrite this equation as a substitution equation for x_j
     - (i.e., divide by a_{ij} & isolate x_j)
   - remove this equation from the system
   - eliminate x_j from the equations remaining in the system, by substituting

(an iteration of this procedure is called a *pivot step for* a_{ij})

2. *back substitute*:
   compute successively the values of $x_n,\ x_{n-1}, \ldots,\ x_1$,
   finding each from its substitution equation $\square$

Accuracy

avoid bad round-off error by initially *scaling* the system to get coefficients of similar magnitude

also, choose each pivot element a_{ij} to have largest magnitude (from among all elements $a_{\cdot j}$)
this is called *partial pivotting*
complete pivotting makes the selection from among all elements $a_{\cdot\cdot}$
so the order that variables are eliminated can change

Time

assume 1 arithmetic operation takes time $O(1)$
i.e., assume the numbers in a computation do not grow too big
(although in pathological cases the numbers can grow exponentially (Edmonds, '67))
time $= O(n^3)$
back substitution is $O(n^2)$
theoretically, solving a linear system uses time $O(M(n)) = O(n^{2.38})$
by divide-and-conquer methods of matrix multiplication & inversion

for sparse systems the time is much less, if we preserve sparsity

Preserving Sparsity

if we pivot on a_{ij} we eliminate row i & column j from the remaining system
  but we can change other coefficients from 0 to nonzero
the number of such changes equals the *fill-in*

we can reduce fill-in by choice of pivot element, for example:

let p_i (q_j) be the number of nonzero entries in row i (column j) in the (remaining) system
pivotting on a_{ij} gives fill-in $\le (p_i - 1)(q_j - 1)$
Markowitz's rule: always pivot on a_{ij}, a nonzero element
  that minimizes $(p_i - 1)(q_j - 1)$
this usually keeps the fill-in small

Remark. minimizing the total fill-in is NP-hard

Matrices for Pivotting

permutation matrix – an identity matrix with rows permuted
let $\mathbf{P}$ be an $n \times n$ permutation matrix
  for any $n \times p$ matrix $\mathbf{A}$, $\mathbf{PA}$ is $\mathbf{A}$ with rows permuted the same as $\mathbf{P}$
  e.g., to interchange rows r and s, take $\mathbf{P}$ an identity with rows r and s interchanged

a permutation matrix can equivalently be described as $\mathbf{I}$ with columns permuted
for any $p \times n$ matrix $\mathbf{A}$, $\mathbf{AP}$ is $\mathbf{A}$ with columns permuted as in $\mathbf{P}$

upper triangular matrix – all entries strictly below the diagonal are 0
similarly *lower triangular matrix*

eta matrix (also called "pivot matrix") –
  identity matrix with 1 column (the *eta column*) changed arbitrarily
  as long as its diagonal entry is nonzero (so it's nonsingular)

Remark. the *elementary matrices* are the permutation matrices and the eta's

in the system $\mathbf{Ax} = \mathbf{b}$ pivotting on a_{kk} results in the system $\mathbf{LAx} = \mathbf{Lb}$
  for $\mathbf{L}$ a lower triangular eta matrix whose kth column entries ℓ_{ik} are
  $0\ (i < k), \quad 1/a_{kk}\ (i = k), \quad -a_{ik}/a_{kk}\ (i > k)$

a pivot in the simplex algorithm is similar ($\mathbf{L}$ is eta but not triangular)
  (a Gauss-Jordan pivot)

Gaussian Elimination Using Matrices

the substitution equations form a system $\mathbf{Ux} = \mathbf{b}'$ (1)
$\mathbf{U}$ is upper triangular, diagonal entries $= 1$

if we start with a system $\mathbf{Ax} = \mathbf{b}$ and do repeated pivots, the kth pivot
(i) interchanges the current kth equation with the equation containing the pivot element, i.e., it premultiplies the system by some permutation matrix $\mathbf{P}_k$
(ii) pivots, i.e., premultiplies by a lower triangular eta matrix $\mathbf{L}_k$, where the eta column is k

so the final system (1) has

$\mathbf{U} = \mathbf{L}_n\mathbf{P}_n\mathbf{L}_{n-1}\mathbf{P}_{n-1}\ldots\mathbf{L}_1\mathbf{P}_1\mathbf{A}$, an upper triangular matrix with diagonal entries 1
$\mathbf{b}' = \mathbf{L}_n\mathbf{P}_n\mathbf{L}_{n-1}\mathbf{P}_{n-1}\ldots\mathbf{L}_1\mathbf{P}_1\mathbf{b}$

matrices $\mathbf{U}, \mathbf{L}_i, \mathbf{P}_i$ form a *triangular factorization* for $\mathbf{A}$
if $\mathbf{A}$ is sparse, can usually achieve sparse $\mathbf{U}$ and $\mathbf{L}_i$'s –
the # of nonzeroes slightly more than doubles (Chvátal, p.92)

Remark. an LUP decomposition of a matrix $\mathbf{A}$ writes it as $\mathbf{A} = \mathbf{LUP}$.

Exercise. In an *integral LP* all coefficients in $\mathbf{A}$, $\mathbf{b}$ & $\mathbf{c}$ integers. Its size in bits is measured by this parameter:
$L = (m+1)n + n\log n + \sum\{\,\log|r| : r \text{ a nonzero entry in } \mathbf{A}, \mathbf{b} \text{ or } \mathbf{c}\}$
The following fact is important for polynomial time LP algorithms:

Lemma. *Any bfs of an integral LP has all coordinates rational numbers whose numerator & denominator have magnitude $< 2^L$.*

Prove the Lemma. To do this recall Cramer's Rule for solving $\mathbf{Ax} = \mathbf{b}$, and apply it to our formula for a dictionary (Handout#23).

in the linear systems solved by the revised simplex algorithm

$\mathbf{yB} = \mathbf{c}_B$, $\mathbf{Bd} = \mathbf{A}_{\cdot s}$

$\mathbf{B}$ can be viewed as a product of eta matrices

thus we must solve "eta systems" of the form

$\mathbf{yE}_1 \ldots \mathbf{E}_k = \mathbf{c}$, $\mathbf{E}_1 \ldots \mathbf{E}_k\mathbf{x} = \mathbf{b}$

Why $\mathbf{B}$ *is a Product of Eta Matrices* (Handout #57 gives a slightly different explanation)

let eta matrix $\mathbf{E}_i$ specify the pivot for ith simplex iteration

if the kth basis is $\mathbf{B}$, then $\mathbf{E}_k \ldots \mathbf{E}_1\mathbf{B} = \mathbf{I}$

thus $\mathbf{B} = (\mathbf{E}_k \ldots \mathbf{E}_1)^{-1} = \mathbf{E}_1^{-1} \ldots \mathbf{E}_k^{-1}$

$\mathbf{B}$ is a product of eta matrices, since

the inverse of a nonsingular eta matrix is an eta matrix

(because the inverse of a pivot is a pivot)

Simple Eta Systems

let $\mathbf{E}$ be an eta matrix, with eta column k

To Solve $\mathbf{Ex} = \mathbf{b}$

1. $x_k = b_k/e_{kk}$
2. for $j \neq k$, $x_j = b_j - e_{jk}x_k$

To Solve $\mathbf{yE} = \mathbf{c}$

1. for $j \neq k$, $y_j = c_j$
2. $y_k = (c_k - \sum_{j \neq k} y_j e_{jk})/e_{kk}$

Time

O(dimension of $\mathbf{b}$ or $\mathbf{c}$)

this improves to time O(#of nonzeros in the eta column) if

(i) we can overwrite $\mathbf{b}$ ($\mathbf{c}$) & change it to $\mathbf{x}$ ($\mathbf{y}$)

both $\mathbf{b}$ & $\mathbf{c}$ are stored as arrays

(ii) $\mathbf{E}$ is stored in a sparse data structure, e.g., a list of nonzero entries in the eta column

e_{kk} is stored first, to avoid 2 passes

General Systems

basic principle: order multiplications to work with vectors rather than matrices

equivalently, work from the outside inwards

let $\mathbf{E}_1, \ldots, \mathbf{E}_k$ be eta matrices

To Solve $\mathbf{E}_1 \ldots \mathbf{E}_k\mathbf{x} = \mathbf{b}$

write the system as $\mathbf{E}_1(\ldots(\mathbf{E}_k\mathbf{x})) = \mathbf{b}$ & work left-to-right:

solve $\mathbf{E}_1\mathbf{b}_1 = \mathbf{b}$ for unknown $\mathbf{b}_1$

then solve $\mathbf{E}_2\mathbf{b}_2 = \mathbf{b}_1$ for unknown $\mathbf{b}_2$

etc., finally solving $\mathbf{E}_k\mathbf{b}_k = \mathbf{b}_{k-1}$ for unknown $\mathbf{b}_k = \mathbf{x}$

To Solve $\mathbf{y}\mathbf{E}_1 \ldots \mathbf{E}_k = \mathbf{c}$

write as $((\mathbf{y}\mathbf{E}_1)\ldots)\mathbf{E}_k = \mathbf{c}$ & work right-to-left:

solve $\mathbf{c}_k\mathbf{E}_k = \mathbf{c}$ for $\mathbf{c}_k$
then solve $\mathbf{c}_{k-1}\mathbf{E}_{k-1} = \mathbf{c}_k$ for $\mathbf{c}_{k-1}$
etc., finally solving $\mathbf{c}_1\mathbf{E}_1 = \mathbf{c}_2$ for $\mathbf{c}_1 = \mathbf{y}$

Time

using sparse data structures time $= O(\text{total \# nonzeros in all } k \text{ eta columns})$
$+O(n)$ if we cannot destroy $\mathbf{b}$ ($\mathbf{c}$)

An Extension

let $\mathbf{U}$ be upper triangular with diagonal entries 1

To Solve $\mathbf{U}\mathbf{E}_1 \ldots \mathbf{E}_k\mathbf{x} = \mathbf{b}$
Method #1:

solve $\mathbf{U}\mathbf{b}_1 = \mathbf{b}$ for $\mathbf{b}_1$ (by back substitution)
then solve $\mathbf{E}_1 \ldots \mathbf{E}_k\mathbf{x} = \mathbf{b}_1$

Method #2 (more uniform):

for $j = 1, \ldots, n$, let $\mathbf{U}_j$ be the eta matrix whose jth column is $\mathbf{U}_{\cdot j}$
Fact: $\mathbf{U} = \mathbf{U}_n\mathbf{U}_{n-1} \ldots \mathbf{U}_1$ (note the order)

Verify this by doing the pivots.

Example: $\begin{bmatrix} 1 & a & b \\ 0 & 1 & c \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & b \\ 0 & 1 & c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & a & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

so use the algorithm for a product of eta matrices

for both methods, the time is essentially $O(\text{total \# nonzero coefficients})$

similarly for $\mathbf{y}\mathbf{U}\mathbf{E}_1 \ldots \mathbf{E}_k = \mathbf{c}$

the revised simplex algorithm can handle LPs with huge numbers of variables!

this technique was popularized by the work of Gilmore & Gomory on the cutting stock problem (the decomposition principle, Chvátal Ch.26, uses the same idea on LPs that overflow memory)

1-Dimensional Cutting Stock Problem

arises in production of paper, foil, sheet metal, etc.

Cutting Stock Problem
raw material comes in "raw" rolls of width r
must produce b_i "final" rolls of width w_i, $i = 1, \ldots, m$
each $w_i \le r$
Problem: minimize the number of raws used

(this problem is NP-complete – just 3 finals per raw is the "3 partition problem")

LP Formulation of Cutting Stock Problem

a "cutting pattern" cuts 1 raw into a_i finals of width w_i $(i = 1, \ldots, m)$
plus perhaps some waste
thus a cutting pattern is any solution to
$\sum_{i=1}^m w_i a_i \le r$, a_i a nonnegative integer

form matrix $\mathbf{A}$ having column $A_{\cdot j}$ = (the jth cutting pattern)
let variable x_j = (# of raws that use pattern j)

cutting stock LP: minimize $[1, \ldots, 1]\mathbf{x}$
subject to $\mathbf{Ax} = \mathbf{b}$
$\mathbf{x} \ge \mathbf{0}$

Remarks

1. the definition of cutting pattern allows us to assume $=$ (rather than $\ge$) in the LP constraints
   also it makes all $c_j = 1$; this will be crucial for column generation

2. the LP ignores the constraint that x_j is integral
   in practice, rounding the LP optimum to an integral solution gives a high-quality answer (since rounding increases the cost by $< m$)
   in some applications, fractional x_j's are OK – see Chvátal

3. the LP is hard to solve because the # of variables x_j is huge
   practical values are $m \approx 40$, $r \approx 500$, $50 \le w_i \le 200$; gives $\approx 10^7$ patterns!

4. a good initial solution to the LP is given by the greedy algorithm, "first fit decreasing":
   use the largest final that fits, and proceed recursively

Delayed Column Generation

this technique adapts the revised simplex algorithm so it can handle
potentially unlimited numbers of variables, if they have a nice structure

to understand the method recall that
the Entering Variable Step seeks a (nonbasic) variable x_j with positive current cost $\overline{c}_j$,
where $\overline{c}_j = c_j - \mathbf{y}A_{\cdot j}$

we implement the Entering Variable Step using a subroutine for the following *auxiliary problem*:

check if current solution $\mathbf{x}$ & and dual variables $\mathbf{y}$ are optimal, i.e., $\mathbf{yA} \geq \mathbf{c}$
if not, find a (nonbasic) column $\mathbf{A}_{\cdot j}, c_j$ with $\mathbf{yA}_{\cdot j} < c_j$

usually we do both parts by maximizing $c_j - \mathbf{yA}_{\cdot j}$

solving the auxiliary problem allows us to complete the simplex iteration:
use the variable returned x_j as the entering variable
use $\mathbf{A}_{\cdot j}$ in the Leaving Variable Step
use c_j in the next Entering Variable Step (for $\mathbf{c}_B$)

thus we solve the LP without explicitly generating $\mathbf{A}$!

The Knapsack Problem

the knapsack problem is to pack a 1-dimensional knapsack with objects of given types,
maximizing the value packed:

maximize $z = \sum_{i=1}^{m} c_i x_i$
subject to $\sum_{i=1}^{m} a_i x_i \leq b$
$x_i \geq 0$, integral $\qquad (i = 1, \ldots, m)$

all c_i & a_i are positive (not necessarily integral)

the knapsack problem
(*i*) is the simplest of ILPs, & a common subproblem in IP
(*ii*) is NP-complete
(*iii*) can be solved efficiently in practice using branch-and-bound
(*iv*) solved in pseudo-polynomial time ($O(mb^2)$) by dynamic programming, for integral a_i's

The Cutting Stock Auxiliary Problem is a Knapsack Problem

for an unknown cutting pattern a_i
(satisfying $\sum_{i=1}^{m} w_i a_i \le r$, a_i nonnegative integer)
we want to maximize $-1 - \sum_{i=1}^{m} y_i a_i$
a pattern costs -1 when we formulate the cutting stock problem as a maximization problem
equivalently we want to maximize $z = \sum_{i=1}^{m} (-y_i) a_i$
can assume $a_i = 0$ if $y_i \ge 0$
this gives a knapsack problem

let z^* be the maximum value of the knapsack problem
$z^* \le 1 \implies$ current cutting stock solution is optimum
$z^* > 1$ gives an entering column $\mathbf{A}_{\cdot s}$

note that our column generation technique amounts to using the largest coefficient rule
experiments indicate this rule gives fewer iterations too!

A Clarification
Chvátal (pp.199–200) executes the simplex algorithm for a minimization problem
i.e., each entering variable is chosen to have negative cost
so his duals are the negatives of those in this handout

Exercises.
1. Prove the above statement, i.e., executing the simplex to minimize z gives current costs & duals that are the negatives of those if we execute the simplex to maximize $-z$.

2. Explain why the entire approach fails in a hypothetical situation where different cutting patterns can have different costs.

branch-and-bound is a method of solving problems by "partial enumeration"
  i.e., we skip over solutions that can't possibly be optimal
invented and used successfully for the Travelling Salesman Problem

in general, a *branch-and-bound search* for a maximum maintains
  M, the value of the maximum solution seen so far
  & a partition of the feasible region into sets S_i
    each S_i has an associated upper bound β_i for solutions in S_i

repeatedly choose i with largest β_i
if $\beta_i \le M$ stop, M is the maximum value
otherwise search for an optimum solution in S_i & either
  find an optimum & update M,
  or split S_i into smaller regions S_j, each with a lower upper bound β_j

Example
consider the *asymmetric TSP*
  the same problem as Handout#1, but we don't assume $c_{ij} = c_{ji}$

asymmetric TSP is this ILP:

$$\begin{array}{lllll}
\text{minimize } z = \sum_{i,j} c_{ij}x_{ij} & & & & \\
\text{subject to} & \sum_{j\ne i} x_{ji} & = 1 & i = 1,\ldots,n & \text{(enter each city once)} \\
 & \sum_{j\ne i} x_{ij} & = 1 & i = 1,\ldots,n & \text{(leave each city once)} \\
 & x_{ij} & \in \{0,1\} & i,j = 1,\ldots,n & \\
 & \sum_{ij\in S} x_{ij} & \le |S|-1 & \emptyset \subset S \subset \{1,\ldots,n\} & \text{(no subtours)}
\end{array}$$

Exercise. Show that the subtour elimination constraints are equivalent to

$$\sum_{i\in S, j\notin S} x_{ij} \ge 1, \qquad \emptyset \subset S \subset \{1,\ldots,n\}$$

for this ILP, as well as for its LP relaxation.

dropping the last line of the ILP gives another ILP, *the assignment problem*
  (see Handout#45,p.2)

  the assignment problem can be solved efficiently (time $O(n^3)$)
  its optimum solution is a lower bound on the TSP solution
  (since it's a relaxation of TSP)

in the following code each partition set S_i is represented by an assignment problem $\mathcal{P}$
    $S_i = \{$ all possible assignments for $\mathcal{P}$ $\}$

$M \leftarrow \infty$ /* M will be the smallest tour cost seen so far */
repeat the following until all problems are examined:
  choose an unexamined problem $\mathcal{P}$
    $\mathcal{P}$ is always an assignment problem
    initially the only unexamined problem is the assignment version of the given TSP
  let α be the optimum cost for assignment problem $\mathcal{P}$
  if $\alpha < M$ **then**
    if the optimum assignment is a tour, $M \leftarrow \alpha$
    otherwise choose an unfixed variable x_{ij} & create 2 new (unexamined) assignment problems:
      the 1st problem is $\mathcal{P}$, fixing $x_{ij} = 0$
      the 2nd problem is $\mathcal{P}$, fixing $x_{ij} = 1$, $x_{ji} = 0$
      /* possibly other $x_{\cdot i}$'s can be zeroed */
  else /* $\alpha \ge M$ */ discard problem $\mathcal{P}$

Remarks.

1. to fix $x_{ij} = 1$, delete row i & column j from the cost matrix
   to fix $x_{ij} = 0$, set $c_{ij} = \infty$
   in both cases we have a new assignment problem

2. the assignment algorithm can use the previous optimum as a starting assignment
   time $O(n^2)$ rather than $O(n^3)$

3. choosing the branching variable x_{ij} (see Exercise for details):
   x_{ij} is chosen as the variable having $x_{ij} = 1$ such that
   setting $x_{ij} = 0$ gives the greatest increase in the dual objective
   sometimes this allows the $x_{ij} = 0$ problem to be pruned before it is solved

Exercise. (a) Show the assignment problem (Handout#45) has dual problem

$$\begin{array}{ll} \text{maximize } z = \sum_{i=1}^n u_i + \sum_{j=1}^n v_j & \\ \text{subject to} \quad u_i + v_j \le c_{ij} & i, j = 1, \ldots, n \end{array}$$

(b) Let z^* be the cost of an optimum assignment. Let u_i, v_j be optimum duals. Show that any feasible assignment with $x_{ij} = 1$ costs $\ge z^* + c_{ij} - u_i - v_j$.

(c) The b-&-b algorithm branches on x_{ij} where $x_{ij} = 1$ and ij maximizes

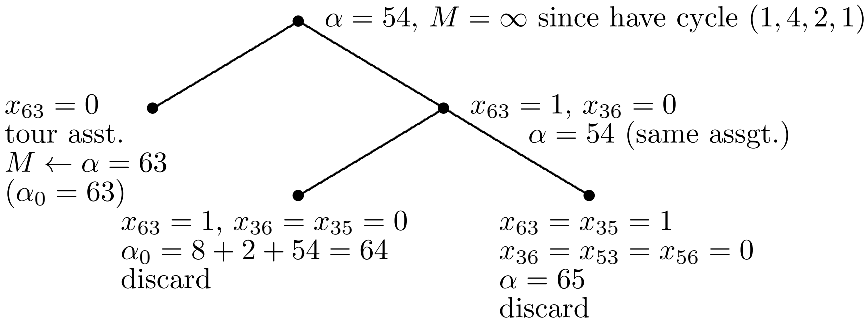
$$\alpha_0 = \min\{c_{ik} - u_i - v_k : k \ne j\} + \min\{c_{kj} - u_k - v_j : k \ne i\} + z^*.$$

Show this choice allows us to discard the $x_{ij} = 0$ problem if $\alpha_0 \ge M$. *Hint.* Use what you learned in (b).

Example Execution

		j						
		1	2	3	4	5	6	u_i
	1	∞	27	43	**16**	30	26	15
	2	**7**	∞	16	1	30	25	0
i	3	20	13	∞	35	**5**	9	0
	4	21	**16**	25	∞	18	18	11
	5	12	46	27	48	∞	**5**	5
	6	23	5	**5**	9	5	∞	0
	v_j	7	5	5	1	5	0	

Cost matrix c_{ij} for given TSP
& optimum assignment (boldface; cost 54)
& optimum duals u_i, v_j (sum 54)



The optimum tour cost is 63.

the b-&-b algorithm has enjoyed success because
the optimum assignment cost is often close to the optimum TSP cost
e.g., in 400 randomly generated asymmetric TSP instances with $50 \le n \le 250$,
the optimum assignment cost averaged $> 99\%$ of the optimum TSP cost
with the bound getting better as n increased (*The Travelling Salesman Problem*, 1985)

we might expect ≈ 1 in n assignments to be a tour
since there are $(n-1)!$ tours,
& the number of assignments (i.e., the number of “derangements”) is $\approx n!/e$
we don’t expect good performance in symmetric problems,
since a cheap edge ij will typically match both ways, ij and ji

Branch-and-Bound Algorithm for Knapsack

recall the Knapsack Problem from last handout:
maximize $z = \sum_{i=1}^{m} c_i x_i$
subject to $\sum_{i=1}^{m} a_i x_i \le b$
$x_i \ge 0$, integral $\qquad (i = 1, \ldots, m)$

like any other b-&-b algorithm, we need
a simple but accurate method to upperbound z

order the items by per unit value, i.e., assume
$c_1/a_1 \ge c_2/a_2 \ge \ldots \ge c_m/a_m$
we’d fill the knapsack completely with item 1, if there were no integrality constraints

can upperbound z for solutions having the first k variables $x_1, \ldots, x_k$ fixed:
$z \le \sum_1^k c_i x_i + (c_{k+1}/a_{k+1})(b - \sum_1^k a_i x_i) = \beta = \beta(x_1, \ldots, x_k)$

since $\sum_{k+1}^{m} c_i x_i \le (c_{k+1}/a_{k+1}) \sum_{k+1}^{m} a_i x_i$

Remarks

1. $\beta(x_1, \ldots, x_k)$ is increasing in x_k
   since increasing x_k by 1 changes β by $c_k - (c_{k+1}/a_{k+1})a_k \geq 0$

2. if all c_i are integral, $z \leq \lfloor \beta \rfloor$

Example. A knapsack can hold $b = 8$ pounds. The 2 densest items have these parameters:

i	1	2
c_i/a_i	2	1
a_i	3	1.1

including 2 items of type 1 gives profit $2 \times 3 \times 2 = 12$
including 1 item of type 1 gives profit $\leq 1 \times 3 \times 2 + 1 \times (8 - 3) = 11$
    regardless of parameters for items $3, 4, \ldots$
    $11 < 12 \implies$ reject this possiblity
including no type 1 items is even worse, profit $\leq 1 \times 8 = 8$

for knapsack a good heuristic is to choose the values x_i by being *greedy*:
initial solution:
    $x_1 \leftarrow \lfloor b/a_1 \rfloor$ (the greatest possible number of item 1)
    $x_2 \leftarrow \lfloor (b - a_1 x_1)/a_2 \rfloor$ (the greatest possible number of item 2, given x_1)
    etc.

Knapsack Algorithm

maintain M as largest objective value seen so far
examine every possible solution, *skipping over* solutions with $\beta \leq M$

set $M = -\infty$ and execute `search`(1)

procedure `search`(k); /* sets x_k, given $x_1, \ldots, x_{k-1}$ */
set x_k greedily (as above), updating β;
if $k = m$ **then** $M \leftarrow \max\{M, z\}$
else repeat {
    `search`($k + 1$);
    decrease x_k by 1, updating β;
    until $x_k < 0$ or $\beta \leq M$; /* by Remark 1 */ }

Remark. Chvátal's algorithm prunes more aggressively, each time β is computed

in practice variables usually have both lower and upper bounds
the simple nature of these constraints motivates our definition of general form

assume, wlog, each x_j has lower bound $\ell_j \in \mathbf{R} \cup \{-\infty\}$ & upper bound $u_j \in \mathbf{R} \cup \{+\infty\}$

forming column vectors ℓ & $\mathbf{u}$ we get this *General Form LP*:

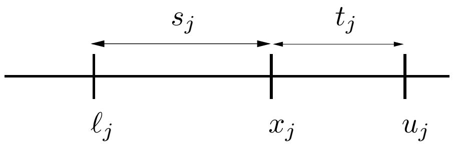
maximize $\mathbf{cx}$
subject to $\mathbf{Ax} = \mathbf{b}$
$\ell \le \mathbf{x} \le \mathbf{u}$

Notation
m = (# of equations); n = (# of variables)
a *free* variable has bounds $-\infty$ and $+\infty$

Converting General Form to Standard Equality Form

replace each variable x_j by 1 or 2 nonnegative variables:

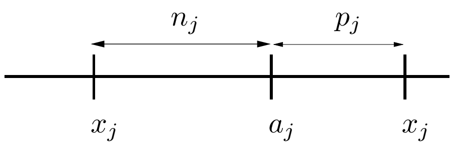
Case 1: x_j has 2 finite bounds.



replace x_j by 2 slacks $s_j, t_j \ge 0$
eliminate x_j by substituting $x_j = \ell_j + s_j$
add constraint $s_j + t_j = u_j - \ell_j$

Case 2: x_j has 1 finite bound.
replace x_j by s_j = (the slack in x_j's bound)
$s_j \ge 0$
eliminate x_j by substituting $x_j = \ell_j + s_j$ or $x_j = u_j - s_j$

Case 3: x_j free, i.e., no finite bounds.



replace x_j by $p_j, n_j \ge 0$
eliminate x_j by substituting $x_j = p_j - n_j$
more generally $x_j = a_j + p_j - n_j$ for some constant a_j

unfortunately the transformation increases m, n

the transformed LP has special structure:

consider an x_j bounded above and below (this increases m)

1. any basis contains at least one of s_j, t_j
   Proof 1. s_j or t_j is nonzero $\square$
   (valid if $\ell_j < u_j$)

   Proof 2. the constraint $s_j + t_j =$ (constant) gives row $0 \ldots 0\ 1\ 1\ 0 \ldots 0$ in $\mathbf{A}$
   so $\mathbf{B}$ contains 1 or both columns s_j, t_j $\square$

2. only s_j basic $\Longrightarrow x_j = u_j$; only t_j basic $\Longrightarrow x_j = \ell_j$

so a basis still has, in some sense, only m variables:
x_j adds a constraint, but also puts a "meaningless" variable into the basis

this motivates Dantzig's *method of upper bounding* discussed in Handout#30
it handles lower & upper bounds without increasing problem size m, n
most LP codes use this method

starting with a General Form LP,

maximize $\mathbf{cx}$
subject to $\mathbf{Ax} = \mathbf{b}$
$\ell \le \mathbf{x} \le \mathbf{u}$

we rework our definitions and the simplex algorithm with lower & upper bounds in mind

basis – list of m columns B, with $\mathbf{A}_B$ nonsingular
basic solution – vector $\mathbf{x} \ni$
(i) $\mathbf{Ax} = \mathbf{b}$
(ii) $\exists$ basis $B \ni j$ nonbasic & nonfree $\Longrightarrow x_j \in \{\ell_j, u_j\}$

Remarks

1. 1 basis can be associated with > 1 basic solution (see Chvátal, p.120)

   a degenerate basic solution has a basic variable equal to its lower or upper bound
   a nondegenerate basic solution has a unique basis if there are no free variables

2. in a *normal basic solution*, any nonbasic free variable equals zero
   running the simplex algorithm on the LP transformed as in previous handout
   only produces normal bfs's
   so non-normal bfs's are unnecessary
   but non-normal bfs's may be useful for initialization (see below)

feasible solution – satisfies all constraints

the simplex algorithm works with a basis B and the usual dictionary relations,

$$\mathbf{x}_B = \mathbf{B}^{-1}\mathbf{b} - \mathbf{B}^{-1}\mathbf{A}_N\mathbf{x}_N, \quad z = \mathbf{c}_B\mathbf{B}^{-1}\mathbf{b} + (\mathbf{c}_N - \mathbf{c}_B\mathbf{B}^{-1}\mathbf{A}_N)\mathbf{x}_N$$

in general $\mathbf{x}^*_B \ne \mathbf{B}^{-1}\mathbf{b}, \quad z^* \ne \mathbf{c}_B\mathbf{B}^{-1}\mathbf{b}$
so our algorithm must maintain the current value of $\mathbf{x}$, denoted $\mathbf{x}^*$ ($\mathbf{x}^*_B$ & $\mathbf{x}^*_N$)

The Simplex Algorithm with Upper Bounding

let B be a basis with corresponding basic solution $\mathbf{x}^*$

Pivot Step

changes the value of some nonbasic x_s
from x^*_s to $x^*_s + \delta$, where δ is positive *or negative*
basic variables change from $\mathbf{x}^*_B$ to $\mathbf{x}^*_B - \delta\mathbf{d}$
objective value changes from z^* to $z^* + (c_s - \mathbf{yA}_{\cdot s})\delta$
as in the revised simplex, $\mathbf{d} = \mathbf{B}^{-1}\mathbf{A}_{\cdot s}$, $\mathbf{y} = \mathbf{c}_B\mathbf{B}^{-1}$

Entering Variable Step

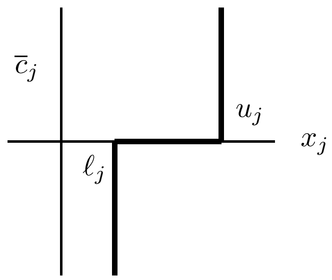
as in revised simplex, find $\mathbf{y} = \mathbf{c}_B \mathbf{B}^{-1}$ & choose entering x_s

2 possibilities for x_s will increase z:

(i) $c_s > \mathbf{y}\mathbf{A}_{\cdot s}$ and $x_s < u_s$ – increase x_s (from its current value ℓ_s, if nonfree)
(ii) $c_s < \mathbf{y}\mathbf{A}_{\cdot s}$ and $x_s > \ell_s$ – decrease x_s (from its current value u_s, if nonfree)
this is the new case

Fact. no variable satisfies (i) or (ii) $\implies B$ is optimal

our "optimality check", i.e. no variable satisfies (i) or (ii), amounts to saying
every variable is "in-kilter", i.e., it is on the following "kilter diagram":



A missing bound eliminates a vertical line from the kilter diagram, e.g., the diagram is the x-axis for a free variable.

Proof.
consider the current bfs $\mathbf{x}^*_B$ and an arbitrary fs $\mathbf{x}$, with objective values z^*, z respectively
from the "dictionary", $z^* - z = (\mathbf{c}_N - \mathbf{c}_B\mathbf{B}^{-1}\mathbf{A}_N)(\mathbf{x}^*_N - \mathbf{x}_N)$
hypothesis implies each term of the inner product is $\ge 0 \implies z^*$ is maximum $\square$

Leaving Variable Step

constraints on the pivot:

(i) $\ell_B \le \mathbf{x}^*_B - \delta\mathbf{d} \le \mathbf{u}_B$
(ii) $\ell_s \le x^*_s + \delta \le u_s$

(half these inequalities are irrelevant)

Case 1: an inequality of (i) *is binding.*
the corresponding variable x_r leaves the basis
the new x_r equals ℓ_r or u_r

Case 2: an inequality of (ii) *is binding.*
basis B stays the same but the bfs changes
x_s changes from one bound (ℓ_s or u_s) to the other

code so ties for the binding variable are broken in favor of this case since it involves less work

Case 3: no binding inequality
LP is unbounded as usual

Other Issues in the Simplex Algorithm

The Role of Free Variables x_j

1. if x_j ever becomes basic, it remains so (see Case 1)
2. if x_j starts out nonbasic, it never changes value until it enters the basis (see Pivot Step)

so non-normal free variables can only help in initialization

a bfs $\mathbf{x}$ is *degenerate* if some basic x_j equals ℓ_j or u_j
degenerate bfs's may cause the algorithm to cycle; avoid by smallest-subscript rule
to understand this think of s_j and t_j has having consecutive subscripts

Initialization

if an initial feasible basis is not obvious, use two-phase method

Phase 1: introduce a *full artificial basis* – artificial variable $v_i,\ i = 1, \ldots, m$
set x_j arbitrarily if free, else to ℓ_j or u_j
multiply ith constraint by -1 if $\mathbf{A}_{i\cdot}\,\mathbf{x} \geq b_i$

Phase 1 LP:

$$\begin{array}{rl} & \text{minimize } \mathbf{1v} \qquad (\mathbf{1} = \text{row vector of } m \text{ 1's}) \\ \text{subject to} & \mathbf{Ax} + \mathbf{Iv} = \mathbf{b} \\ & \ell \leq \mathbf{x} \leq \mathbf{u} \\ & \mathbf{0} \leq \mathbf{v} \end{array}$$

a basis with $\mathbf{v} = \mathbf{0}$ gives a bfs for the original LP:
drop all nonbasic artificial variables
for each basic artificial variable v_i add constraints $0 \leq v_i \leq 0$

a non-normal bfs can help in initialization:

e.g., consider the LP maximize x_1 such that

$$\begin{aligned} x_1 \qquad\quad + x_3 &= -1 \\ x_1 + 3x_2 + 4x_3 &= -13 \\ x_1, x_2 &\geq 0 \end{aligned}$$

equivalently $x_1 = -1 - x_3, \quad x_2 = -x_3 - 4$
so take $x_3 = -4$ and initial basis $(1, 2)$, $x_1 = 3,\ x_2 = 0$
Phase 1 not needed, go directly to Phase 2

the method of *generalized upper bounding* (Chvátal, Ch. 25)
adapts the simplex algorithm for constraints of the form
$\sum_{j \in S} x_j = b$
each x_j appearing in ≤ 1 such constraint

BFSs in General LPs

a standard form LP, converted to a dictionary, automatically has a basis
formed by the slacks – not necessarily feasible

this general LP

$$\begin{aligned} \text{maximize } & x_1 \\ \text{subject to } x_1 + x_2 &= 1 \\ 2x_1 + 2x_2 &= 2 \\ x_1, x_2 &\geq 0 \end{aligned}$$

has optimum solution $x_1 = 1$, $x_2 = 0$ but no basis!
since $\begin{bmatrix} 1 & 1 \\ 2 & 2 \end{bmatrix}$ is singular

Phase 1, using artificial variables v_1, v_2, terminates successfully with LP

$$\begin{aligned} &\text{minimize } 3v_1 \\ &\text{subject to } x_1 = 1 - x_2 - v_1 \\ &\qquad v_2 = \qquad 2v_1 \\ &\qquad x_i, v_i \geq 0 \end{aligned}$$

to proceed to Phase 2 we drop v_1
we could safely drop v_2 and its constraint $v_2 = 0$
this would give us a basis
intuitively this corresponds to eliminating the redundant constraint
we'll show how to eliminate redundancy in general, & get a basis

consider a general form LP $\mathcal{L}$:
maximize $\mathbf{cx}$ subject to $\mathbf{Ax} = \mathbf{b}$, $\ell \leq \mathbf{x} \leq \mathbf{u}$

$\mathcal{L}$ has a basis $\iff$ some m columns B have $\mathbf{A}_B$ nonsingular
$\iff$ $\mathbf{A}$ has full row rank

the *row rank* of an $m \times n$ matrix $\mathbf{A}$ is the maximum # of linearly independent rows
similarly for *column rank*
(row rank of $\mathbf{A}$) = (column rank of $\mathbf{A}$) = the *rank* of $\mathbf{A}$

to prove (row rank) = (column rank) it suffices to show
$\mathbf{A}$ has full row rank $\implies$ it has m linearly independent columns

Proof.
let $\mathbf{B}$ be a maximal set of linearly independent columns
so any other column $\mathbf{A}_{.j}$ is dependent on $\mathbf{B}$, i.e., $\mathbf{B}\mathbf{x}_j = \mathbf{A}_{.j}$
the rows of $\mathbf{B}$ are linearly independent:
$\mathbf{rB} = \mathbf{0} \implies \mathbf{rA}_{.j} = \mathbf{rBx}_j = \mathbf{0}$. so $\mathbf{rA} = \mathbf{0}$. this makes $\mathbf{r} = \mathbf{0}$
thus $\mathbf{B}$ has $\geq m$ columns $\square$

Eliminating Artificial Variables & Redundant Constraints

A Simple Test for Nonsingularity

define
$\mathbf{A}$: a nonsingular $n \times n$ matrix
$\mathbf{a}$: a length n column vector
$\mathbf{A}'$: $\mathbf{A}$ with column n replaced by $\mathbf{a}$
$\mathbf{r}$: the last row of $\mathbf{A}^{-1}$, i.e.,
a length n row vector $\ni$ $\mathbf{rA} = (0, \ldots, 0, 1)$

Lemma. $\mathbf{A}'$ *is nonsingular* $\iff \mathbf{ra} \neq 0$.

Proof.
we prove the 2 contrapositives (by similar arguments)

$\mathbf{ra} = 0 \implies \mathbf{rA}' = \mathbf{0} \implies \mathbf{A}'$ singular

$$
\begin{aligned}
\mathbf{A}' \text{ singular } &\implies \text{for some row vector } \mathbf{s} \neq \mathbf{0},\ \mathbf{sA}' = \mathbf{0} \\
&\implies \mathbf{sA} = (0, \ldots, 0, x),\ x \neq 0 \text{ (since } \mathbf{A} \text{ is nonsingular)} \\
&\implies \mathbf{s} = x\mathbf{r} \\
&\implies \mathbf{ra} = 0 \quad \square
\end{aligned}
$$

this gives an efficient procedure to get a bfs for an LP –

1. solve the Phase 1 LP
   get a feasible basis $\mathbf{B}$, involving artificial variables v_i

2. eliminate artificials from $\mathbf{B}$ as follows:
   for each basic artificial v_i **do**
   let v_i be basic in the kth column
   solve $\mathbf{rB} = \mathbf{I}_{k.}$ /*$\mathbf{r}$ is the vector of the lemma */
   replace v_i in $\mathbf{B}$ by a (nonbasic) variable x_j $\ni$ $\mathbf{rA}_{.j} \neq 0$, if such a j exists

the procedure halts with a basis (possibly still containing artificials), by the lemma
have same bfs (i.e., each new x_j equals its original value)

Procedure to Eliminate Redundant Constraints

let $R = \{k : v_k \text{ remains in } \mathbf{B} \text{ after the procedure}\}$
"R" stands for redundant
form LP $\mathcal{L}'$ by dropping the constraints for R

1. $\mathcal{L}'$ is equivalent to $\mathcal{L}$, i.e., they have the same feasible points

Proof.
take any $k \in R$
for simplicity assume v_k is basic in row k

$$\text{e.g., for } m = 5,\ |R| = 3, \mathbf{B} \text{ is } \begin{bmatrix} 1 & 0 & 0 & \dots \\ 0 & 1 & 0 & \dots \\ 0 & 0 & 1 & \dots \\ 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & \dots \end{bmatrix}$$

let $\mathbf{r}$ be the row vector used in the procedure for v_k
$r_k = 1$; $r_j = 0$ for $j \in R - k$
$\mathbf{rA} = \mathbf{0} \implies$ the kth row is a linear combination of the rows of $\mathcal{L}'$ □

2. $\mathcal{L}'$ has basis $B' = B - R$

Proof.
in $\mathbf{B}$, consider the rows for constraints of $\mathcal{L}'$
any entry in a column of R is 0
$\implies$ these rows are linearly independent when the columns of R are dropped
$\implies$ these rows form a basis of $\mathcal{L}'$ □

Generalized Fundamental Theorem of LP. *Consider any general LP $\mathcal{L}$.*

(i) Either $\mathcal{L}$ has an optimum solution
or the objective is unbounded or the constraints are infeasible.

Suppose $\mathbf{A}$ has full row rank.

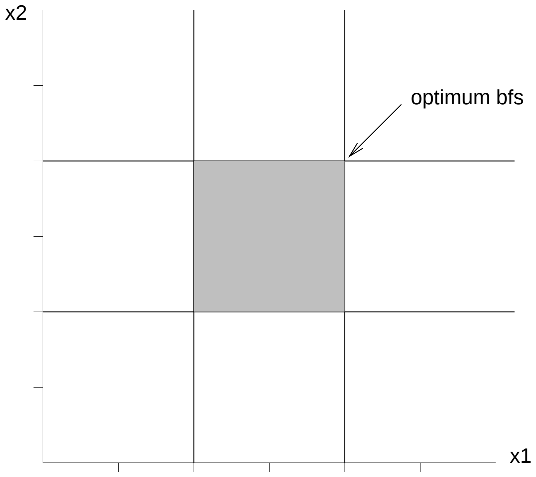
(ii) $\mathcal{L}$ feasible $\implies$ there is a normal bfs.

(iii) $\mathcal{L}$ has an optimum $\implies$ the optimum is achieved by a normal bfs. □

Proof.

for (i), run the general simplex algorithm (2-Phase)
for (ii), initialize Phase 1 with a normal bfs
it halts with a normal bfs
eliminate *all* artificial variables using the above procedure □

Example
$\mathcal{L}$: maximize $x_1 + x_2$ subject to $2 \le x_1 \le 4$, $2 \le x_2 \le 4$
this LP has empty $\mathbf{A}$, which has full row rank!



Extended Fundamental Theorem (Chvátal p.242):
If $\mathbf{A}$ *has full row rank & $\mathcal{L}$ is unbounded, there is a basic feasible direction with positive cost.*

$\mathbf{w}$ is a *feasible direction* if $\mathbf{Aw} = \mathbf{0}$, $w_j < 0$ only when $\ell_j = -\infty$ & $w_j > 0$ only when $u_j = \infty$

$\mathbf{w}$ is a *basic feasible direction* if $\mathbf{A}$ has a basis $\ni$
exactly 1 nonbasic variable w_j is nonzero, and $w_j = \pm 1$

the general simplex algorithm proves the extended theorem:
$x^* - \delta\mathbf{d}$ is feasible

Certificates of Infeasiblity

let $\mathcal{I}$ be a system of inequalities $\mathbf{Ax} \le \mathbf{b}$

Theorem. *$\mathcal{I}$ is infeasible $\Longleftrightarrow$*
the contradiction $0 \le -1$ can be obtained as a linear combination of constraints.

Proof.
consider this primal-dual pair:

primal	*dual*
maximize $\mathbf{0x}$	minimize $\mathbf{yb}$
subject to $\mathbf{Ax} \le \mathbf{b}$	subject to $\mathbf{yA} = \mathbf{0}$
	$\mathbf{y} \ge \mathbf{0}$

(the primal is $\mathcal{I}$ with a constant objective function 0)

$\mathcal{I}$ infeasible $\Longrightarrow$ dual unbounded (since dual is feasible, e.g., $\mathbf{y} = \mathbf{0}$)
$\Longrightarrow$ some feasible $\mathbf{y}$ has $\mathbf{yb} = -1$
i.e., a linear combination of constraints of $\mathcal{I}$ gives $0 \le -1$ $\square$

let $n = (\#$ variables in $\mathcal{I})$

Corollary. *$\mathcal{I}$ is infeasible $\Longleftrightarrow$*
some subsystem of $\le n+1$ constraints is infeasible.

Proof.
first assume $\mathbf{A}$ has full column rank
this implies $[\mathbf{A} \mid \mathbf{b}]$ has full column rank
  if not, $\mathbf{b}$ is a linear combination of columns of $\mathbf{A}$, contradicting infeasiblity

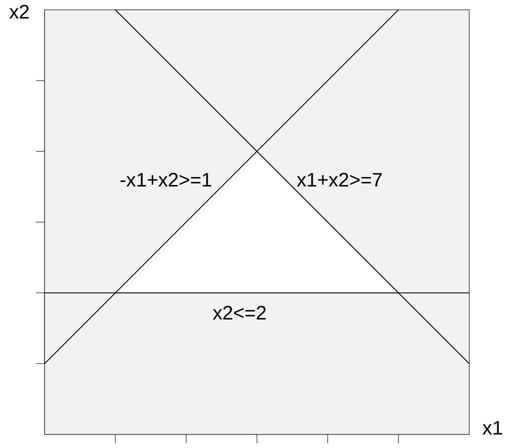
$\mathcal{I}$ infeasible $\Longrightarrow \mathbf{yA} = \mathbf{0}, \quad \mathbf{yb} = -1, \quad \mathbf{y} \ge \mathbf{0}$ is feasible
$\Longrightarrow$ it has a bfs (in the general sense) $\mathbf{y}^*$
  by the Generalized Fundamental Theorem of LP, and full column rank

there are $n+1$ constraints, so $n+1$ basic variables
any nonbasic variable is 0
so $\mathbf{y}^*$ has $\le n+1$ positive variables

now consider a general $\mathbf{A}$
drop columns of $\mathbf{A}$ to form $\mathbf{A}'$ of full column rank, & apply above argument
the multipliers $\mathbf{y}^*$ satisfy $\mathbf{y}^*\mathbf{A}' = 0$
this implies $\mathbf{y}^*\mathbf{A} = 0$ since each dropped column is linearly dependent on $\mathbf{A}'$ $\square$

Remarks

1. the corollary can't be strengthened to n infeasible constraints
   e.g., in this system in variables x_1, x_2, any 2 constraints are feasible:



2. both results extend to allow equality constraints in $\mathcal{I}$
   the proof is unchanged (just messier notation)

3. Chvátal proves both results differently

Inconsistency in the Simplex Algorithm

we'll now show the Phase 1 Infeasibility Proof (Handout #13,p.4) is correct, i.e.,
executing the Phase 1 simplex algorithm on an inconsistent system $\mathcal{I}$,
the final Phase 1 dictionary gives the multipliers of the corollary:

recall the starting dictionary for Phase 1,
with artificial variable x_0 & slacks $x_j,\ j = n+1, \ldots, n+m$:

$$\begin{array}{l} x_{n+i} = b_i - \sum_{j=1}^n a_{ij}x_j + x_0 \qquad (i = 1, \ldots, m) \\ \hline w = \qquad\qquad\qquad\qquad -x_0 \end{array}$$

the final Phase 1 dictionary has objective

$$w = \overline{w} + \sum_j \overline{c}_j x_j$$

where each $\overline{c}_j \le 0$ & (assuming inconsistency) $\overline{w} < 0$

this equation is a linear combination of equations of the starting dictionary
(Lemma 1 of Handout#16)

i.e., setting $y_i = -\overline{c}_{n+i}$, the equation is

$$w = -x_0 + \sum_{i=1}^m y_i(b_i - \sum_{j=1}^n a_{ij}x_j + x_0 - x_{n+i})$$

thus multiplying the ith original inequality by y_i and adding gives

$$\sum_{i=1}^m y_i(\sum_{j=1}^n a_{ij}x_j) \le \sum_{i=1}^m y_i b_i$$

i.e., $\sum_{j=1}^n -\overline{c}_j x_j \le \overline{w}$

each term on the l.h.s. is nonnegative but the r.h.s. is negative, contradiction!

duality gives many other characterizations for feasibility of systems of inequalities
they're called *theorems of alternatives*
  they assert that exactly 1 of 2 systems has a solution
e.g., here's one you already know:
Farkas's Lemma for Gaussian Elimination.
For any $\mathbf{A}$ *and* $\mathbf{b}$*, exactly 1 of these 2 systems is feasible:*
(I) $\mathbf{Ax} = \mathbf{b}$
(II) $\mathbf{yA} = \mathbf{0}, \quad \mathbf{yb} \neq 0$

Example.

$$\begin{aligned} x_1 - x_2 &= 1 \\ -2x_1 + x_2 &= 0 \\ 3x_1 - x_2 &= 1 \end{aligned}$$

  adding twice the 2nd constraint to the other two gives $0 = 2$

Farkas' Lemma. (1902) *For any* $\mathbf{A}$ *and* $\mathbf{b}$*, exactly 1 of these 2 systems is feasible:*
(I) $\mathbf{Ax} = \mathbf{b}, \quad \mathbf{x} \geq \mathbf{0}$
(II) $\mathbf{yA} \geq \mathbf{0}, \quad \mathbf{yb} < 0$

Interpretations:
(*i*) system (I) is infeasible iff it implies the contradiction (nonnegative #) = (negative #)
(*ii*) system (II) is infeasible iff it implies the contradiction (negative #) ≥ 0

Example: the system

$$\begin{aligned} x_1 - x_2 &= 1 \\ 2x_1 - x_2 &= 0 \end{aligned}$$

is inconsistent, since
  $-1\times$ (first constraint) + (2nd constraint)
gives $x_1 = -1$, i.e., (nonnegative #) = (negative #)

Proof.

consider this primal-dual pair:

Primal	*Dual*
maximize $\mathbf{0x}$	minimize $\mathbf{yb}$
subject to $\mathbf{Ax} = \mathbf{b}$	subject to $\mathbf{yA} \geq \mathbf{0}$
$\mathbf{x} \geq \mathbf{0}$	

(I) feasible $\iff$ 0 is the optimum objective value for both primal & dual

$\iff$ (II) infeasible
  for $\Longleftarrow$ note that $\mathbf{y} = \mathbf{0}$ gives dual objective 0 $\square$

Remarks

1. Farkas' Lemma useful in linear, nonlinear and integer programming

   Integer Version:
   For $\mathbf{A}$ and $\mathbf{b}$ integral, exactly 1 of these 2 systems is feasible:

   $\mathbf{Ax} = \mathbf{b}, \quad \mathbf{x} \in \mathbf{Z}^n$

   $\mathbf{yA} \in \mathbf{Z}^n, \quad \mathbf{yb} \notin \mathbf{Z}, \quad \mathbf{y} \in \mathbf{R}^m$

   Example
   consider this system of equations in integral quantities x_i:

   $$2x_1 + 6x_2 + x_3 = 8$$
   $$4x_1 + 7x_2 + 7x_3 = 4$$

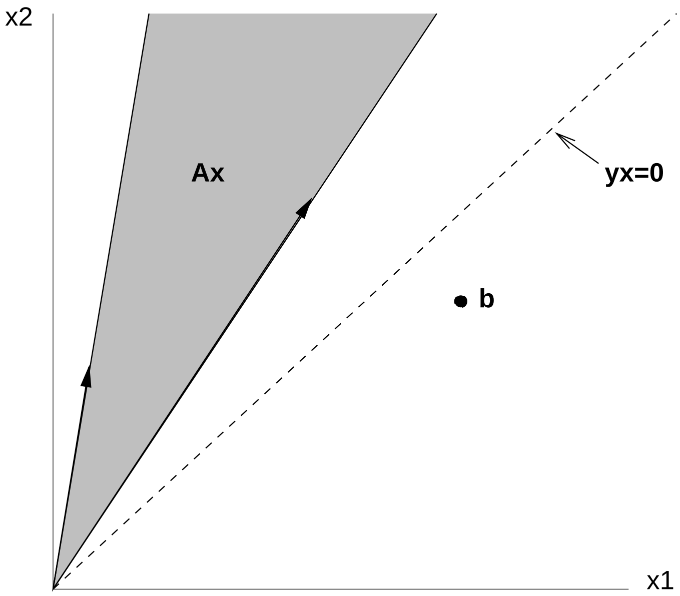
   tripling the 1st equation & adding the 2nd gives the contradiction

   $$10x_1 + 25x_2 + 10x_3 = 28$$

   the corresponding vector for Farkas' Lemma is $y_1 = 3/5$, $y_2 = 1/5$

2. Farkas's Lemma is a special case of the Separating Hyperplane Theorem:
   S a closed convex set in $\mathbf{R}^m$, $\mathbf{b} \in \mathbf{R}^m - S \implies$
   some hyperplane separates $\mathbf{b}$ from S, i.e., $\mathbf{yb} > a$, $\mathbf{ys} \le a$ for all $\mathbf{s} \in S$

   for Farkas, S is the cone generated by the columns of $\mathbf{A}$
   (I) says $\mathbf{b}$ is in the cone, (II) says $\mathbf{b}$ can be separated from it



Farkas's Lemma in requirement space

our last theorem of alternatives deals with strict inequalities

Lemma (Gordan). *For any* $\mathbf{A}$*, exactly 1 of these 2 systems is feasible:*
(I) $\mathbf{Ax} < \mathbf{0}$
(II) $\mathbf{yA} = \mathbf{0}, \quad \mathbf{y} \geq \mathbf{0}, \quad \mathbf{y} \neq \mathbf{0}$

Proof.

consider this primal-dual pair:

Primal	*Dual*
maximize ϵ	minimize $\mathbf{y0}$
subject to $\mathbf{Ax} + \epsilon\mathbf{1} \leq \mathbf{0}$	subject to $\mathbf{yA} = \mathbf{0}$
	$\mathbf{y1} = 1$
	$\mathbf{y} \geq \mathbf{0}$

$\mathbf{1}$ denotes a column vector of 1's

(I) feasible $\Longleftrightarrow$ primal unbounded
$\Longleftrightarrow$ dual infeasible (since primal is feasible, all variables 0)
$\Longleftrightarrow$ (II) infeasible (by scaling $\mathbf{y}$) $\square$

Remarks.

1. Here's a generalization of Gordan's Lemma to nonlinear programming:

Theorem (Fan et al, 1957).
Let $\mathbf{C}$ *be a convex set in* $\mathbf{R}^n$*, and let* $\mathbf{f} : \mathbf{C} \to \mathbf{R}^m$ *be a convex function. Then exactly 1 of these 2 systems is feasible:*
(I) $\mathbf{f}(\mathbf{x}) < \mathbf{0}$
(II) $\mathbf{yf}(\mathbf{x}) \geq \mathbf{0}$ *for all* $\mathbf{x} \in \mathbf{C}$, $\mathbf{y} \geq \mathbf{0}$, $\mathbf{y} \neq \mathbf{0}$

Exercise. Prove Fan's Theorem includes Gordan's as a special case. Begin by taking $\mathbf{f}(\mathbf{x}) = \mathbf{Ax}$. The challenge is to prove $\mathbf{yA} = \mathbf{0}$, as required in Gordan, not $\mathbf{yA} \geq \mathbf{0}$, which looks like what comes out of Fan.

2. Chvátal gives other theorems of alternatives

we can solve an LP by running the simplex algorithm on the dual
the dual simplex algorithm amounts to that but is executed on primal dictionaries

DS Example. we'll show that for the LP

$$\begin{array}{lrl}
\text{maximize } z = & 8x_1+5x_2 & \\
\text{subject to} & x_1+\ x_2 & \le 6 \\
& 9x_1+5x_2 & \le 45 \\
& 3x_1+2x_2 & \le 15 \\
& x_1, x_2 & \ge 0
\end{array}$$

and initial dictionary

$$\begin{array}{rcrl}
x_1 &=& \frac{15}{4} & +\frac{5}{4}\,s_1 - \frac{1}{4}\,s_2 \\
x_2 &=& \frac{9}{4} & -\frac{9}{4}\,s_1 + \frac{1}{4}\,s_2 \\
s_3 &=& -\frac{3}{4} & +\frac{3}{4}\,s_1 + \frac{1}{4}\,s_2 \\
\hline
z &=& \frac{165}{4} & -\frac{5}{4}\,s_1 - \frac{3}{4}\,s_2
\end{array}$$

1 dual simplex pivot gives the optimal dictionary

$$\begin{array}{rcrl}
x_1 &=& 5 & -\frac{2}{3}\,s_2 + \frac{5}{3}\,s_3 \\
x_2 &=& 0 & +s_2 + 3\,s_3 \\
s_1 &=& 1 & -\frac{1}{3}\,s_2 + \frac{4}{3}\,s_3 \\
\hline
z &=& 40 & -\frac{1}{3}\,s_2 - \frac{5}{3}\,s_3
\end{array}$$

Dual Feasible Dictionaries

consider an LP for the standard simplex, and its dual:

Primal	*Dual*
maximize $z = \mathbf{cx}$	minimize $\mathbf{yb}$
subject to $\mathbf{Ax} = \mathbf{b}$	subject to $\mathbf{yA} \ge \mathbf{c}$
$\mathbf{x} \ge \mathbf{0}$	

recall the cost equation in a dictionary: $z = \mathbf{yb} + \overline{\mathbf{c}}_N\mathbf{x}_N$
where $\mathbf{y} = \mathbf{c}_B\mathbf{B}^{-1}$, $\overline{\mathbf{c}}_N = \mathbf{c}_N - \mathbf{yA}_N$

simplex halts when $\overline{\mathbf{c}}_N \le \mathbf{0}$
this is equivalent to $\mathbf{y}$ being dual feasible
show by considering the dual constraints for B & N separately
so call a dictionary *dual feasible* when $\overline{\mathbf{c}}_N \le \mathbf{0}$
e.g., the 2 dictionaries of DS Example

A dictionary that is primal and dual feasible is optimal

Proof #1: this dictionary makes simplex algorithm halt with optimum solution

Proof #2: $\mathbf{x}$ and $\mathbf{y}$ satisfy strong duality

Idea of Dual Simplex Algorithm & Comparison with Standard Simplex

Simplex Algorithm	*Dual Simplex Algorithm*
maintains a primal feasible solution	maintains a dual feasible solution
each iteration increases the objective	each iteration decreases the objective
halts when dual feasibility is achieved	halts when primal feasibility is achieved

why does dual simplex decrease the objective?
to improve the current dictionary we must increase some nonbasic variables
this decreases z (or doesn't change it) by the above cost equation

Sketch of a Dual Simplex Pivot

Example. the optimum dictionary of DS Example results from
a dual simplex pivot on the initial dictionary, with s_1 entering and s_3 leaving

consider a dictionary with coefficients a_{ij}, b_i, c_j
the dictionary is dual feasible (all $c_j \le 0$) but primal infeasible (some b_i are negative)
we want to pivot to a better dual feasible dictionary

i.e., a negative basic variable increases
because a nonbasic variable increases (from 0)

starting pivot row: $x_r = b_r - \sum_{j \in N} a_{rj} x_j$

in keeping with our goal we choose a row with b_r negative
want $a_{rs} < 0$ so increasing x_s increases x_r

new pivot row: $x_s = (b_r/a_{rs}) - \sum_{j \in N'} (a_{rj}/a_{rs}) x_j$

here $N' = N - \{s\} \cup \{r\}, \quad a_{rr} = 1$
note the new value of x_s is positive, the quotient of 2 negative numbers

new cost row: $z = (\text{original } z) + (c_s b_r/a_{rs}) + \sum_{j \in N'} [c_j - (c_s a_{rj}/a_{rs})] x_j \qquad (1)$

here $c_r = 0$
the cost decreases when $c_s < 0$

to maintain dual feasibility, want $c_j \le c_s a_{rj}/a_{rs}$ for all nonbasic j
true if $a_{rj} \ge 0$
so choose s to satisfy $c_j/a_{rj} \ge c_s/a_{rs}$ for all nonbasic j with $a_{rj} < 0$

Example.
in the initial dictionary of DS Example, the ratios for $s = 3$ are $s_1 : \frac{5}{4}/\frac{3}{4} = 5/3, \quad s_2 : \frac{3}{4}/\frac{1}{4} = 3.$
min ratio test $\Longrightarrow s_1$ enters

Standard Dual Simplex Algorithm

let a_{ij}, b_i, c_j refer to the current dictionary, which is dual feasible

Leaving Variable Step
If every $b_i \ge 0$, stop, the current basis is optimum
Choose any (basic) r with $b_r < 0$

Entering Variable Step
If every $a_{rj} \ge 0$, stop, the problem is infeasible
Choose a (nonbasic) s with $a_{rs} < 0$ that minimizes c_s/a_{rs}

Pivot Step
Construct dictionary for the new basis as usual □

Correctness of the Algorithm

Entering Variable Step:
if every $a_{rj} \ge 0$, starting equation for x_r is unsatisfiable
    nonnegative # = negative #

termination of the algorithm:
pivots with $c_s < 0$ decrease z
    pivots with $c_s = 0$ don't change z
finite # bases $\Longrightarrow$ such pivots eventually cause algorithm to halt
    unless it cycles through pivots with $c_s = 0$

a pivot is *degenerate* if $c_s = 0$
a degenerate pivot changes $\mathbf{x}$, but not the cost row ($\mathbf{y}$)
cycling doesn't occur in practice
it can be prevented as in the standard simplex algorithm
alternatively, see Handout#60 □

Remarks

1. the Entering and Leaving Variable Steps are reversed from standard simplex

2. a pivot kills 1 negative variable, but it can create many other negative variables
   e.g., in Chvátal pp. 155–156 the first pivot kills 1 negative variable but creates another
   in fact the total infeasibility (total of all negative variables) increases in magnitude

3. dual simplex allows us to avoid Phase I for blending problems
   the initial dictionary is dual feasible

   in general a variant of the big-M method can be used to initialize the dual simplex

4. the `CPLEX` dual simplex algorithm is particularly efficient
   because of a convenient pivot rule

Revised Dual Simplex Algorithm

as in primal revised simplex maintain

the basis heading & eta factorization of the basis, $x^*_B = \overline{b}$ (current basic values)

in addition maintain the current nonbasic cost coefficients $\overline{c}_N$

$\mathbf{y}$ isn't needed, but the Entering Variable Step must compute *every* $\mathbf{vA}_N$

Leaving Variable Step: same as standard

Entering Variable Step:
we need the dictionary equation for x_r,

$\mathbf{x}_r = \mathbf{I}_{r\cdot}\mathbf{B}^{-1}\mathbf{b} - \mathbf{I}_{r\cdot}\mathbf{B}^{-1}\mathbf{A}_N\mathbf{x}_N$

first solve $\mathbf{vB} = \mathbf{I}_{r\cdot}$

then compute the desired coefficients $\overline{a}_{rj}$, $j \in N$ as $\mathbf{vA}_N$

Pivot Step:
solve $\mathbf{Bd} = \mathbf{A}_{\cdot s}$

to update the basic values $\mathbf{x}^*_B$,

$x^*_s \leftarrow x^*_r/\overline{a}_{rs}$

the rest of the basis is updated by decreasing x^*_B by $x^*_s\mathbf{d}$

use $\mathbf{d}$ to update the eta file

update $\overline{c}_N$ as indicated in (1) $\square$

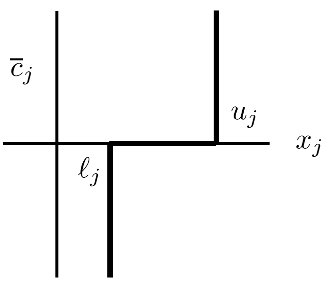
General Dual Simplex

a basic solution $\mathbf{x}$ is *dual feasible* if the cost $\overline{c}_j$ of each nonbasic variable x_j satisfies

$\overline{c}_j > 0 \Longrightarrow x_j = u_j$

$\overline{c}_j < 0 \Longrightarrow x_j = \ell_j$

i.e., each nonbasic variable is "in-kilter" as in Handout#30:



the algorithm maintains the non-basic variables in-kilter
halting when all basic variables are in-kilter, i.e., within their bounds
& otherwise pivotting to bring the leaving basic variable into kilter

details similar to revised dual simplex

postoptimality analysis studies the optimum solution – its robustness and patterns of change
the ease of doing this is an important practical attraction of LP and the simplex algorithm
in contrast postoptimality analysis is very hard for IP

assume we've solved a general LP,

maximize $\mathbf{cx}$ subject to $\mathbf{Ax} = \mathbf{b}$, $\ell \le \mathbf{x} \le \mathbf{u}$

obtaining optimum basis $\mathbf{B}$ and corresponding optimum bfs $\mathbf{x}^*$

sensitivity analysis tells how to solve slightly-changed versions of the problem
e.g., $\mathbf{c}$ changes to $\widetilde{\mathbf{c}}$
$\widetilde{\ }$ will always denotes modified parameters

Cost Changes, $\widetilde{\mathbf{c}}$

$\mathbf{B}$ is a basis for the new problem with $\mathbf{x}^*$ a bfs
so simply restart the revised simplex algorithm, using $\widetilde{\mathbf{c}}$ but no other changes
(standard simplex: must recalculate $\overline{c}$ but that's easy)

Cost Ranging

assuming only 1 cost c_j changes
find the values of c_j for which $\mathbf{B}$ and $\mathbf{x}^*$ are optimal
we'll show the answer is a closed interval – the *interval of optimality* of $\mathbf{B}$ & $\mathbf{x}^*$

recall $z = \mathbf{yb} + (\mathbf{c}_N - \mathbf{yA}_N)\mathbf{x}_N$ where $\mathbf{y} = \mathbf{c}_B\mathbf{B}^{-1}$
the solution is optimum as long as each variable is in-kilter

Case 1. j nonbasic.

$x_j = \ell_j$: $c_j \le \mathbf{yA}_{.j}$ gives the interval of optimality
$x_j = u_j$: $c_j \ge \mathbf{yA}_{.j}$
x_j free: $c_j = \mathbf{yA}_{.j}$ (trivial interval)

Case 2. j basic.

compute the new multipliers for $\mathbf{B}$:
$\widetilde{\mathbf{y}} = \widetilde{\mathbf{c}}_B\mathbf{B}^{-1} = \mathbf{y}^1 c_j + \mathbf{y}^2$, i.e., $\widetilde{\mathbf{y}}$ depends linearly on c_j
using $\widetilde{\mathbf{y}}$, each nonbasic variable gives a lower or upper bound (or both) for c_j, as in Case 1
note the interval of optimality is closed

the optimum objective z^* changes by $\begin{cases} 0 & \text{Case 1} \\ \Delta(c_j)\mathbf{y}^1\mathbf{b} & \text{Case 2} \end{cases}$

Right-hand Side Changes, $\widetilde{\mathbf{b}}$

$\mathbf{B}$ remains a basis
it has a corresponding bfs

$\overline{\mathbf{x}}_N = \mathbf{x}^*_N$

$\overline{\mathbf{x}}_B = \mathbf{B}^{-1}\widetilde{\mathbf{b}} - \mathbf{B}^{-1}\mathbf{A}_N\overline{\mathbf{x}}_N \qquad (*)$

this bfs is dual-feasible
so we start the revised dual simplex algorithm using the above bfs
the eta file and current cost coefficients are available from the primal simplex

r.h.s. ranging is similar, using $(*)$, ℓ & $\mathbf{u}$
changing 1 b_i gives ≤ 2 inequalities per basic variable

more generally suppose in addition to $\widetilde{\mathbf{b}}$ have new bounds $\widetilde{\ell}$, $\widetilde{\mathbf{u}}$

$\mathbf{B}$ remains a basis
define a new bfs $\overline{\mathbf{x}}$ as follows:

for j nonbasic, $\overline{x}_j =$ **if** x_j is free in new problem **then** x^*_j
else if $x^*_j = \ell_j$ **then** $\widetilde{\ell}_j$
else if $x^*_j = u_j$ **then** $\widetilde{u}_j$
else /* x_j was free and is now bound */ a finite bound $\widetilde{\ell}_j$ or $\widetilde{u}_j$

for this to make sense we assume no finite bound becomes infinite
also define $\overline{\mathbf{x}}_B$ by $(*)$

$\overline{\mathbf{x}}$ is dual feasible
hence restart revised dual simplex from $\overline{\mathbf{x}}$

Question. Why does the method fail if a finite bound becomes infinite?

Adding New Constraints

add a slack variable v_i in each new constraint
v_i has bounds $0 \le v_i < \infty$ for an inequality & $0 \le v_i \le 0$ for an equation

extend $\mathbf{B}$ & $\mathbf{x}^*$ to the new system:
add each v_i to the basis
compute its value from its equation

we have a dual-feasible solution ($\text{cost}(v_i) = 0$) so now use the dual simplex algorithm
refactor the basis since the eta file changes

DS Example (Handout#34) cont'd.
we solve the LP

$$\begin{array}{lrl}
\text{maximize } z = & 8x_1 + 5x_2 & \\
\text{subject to} & x_1 + x_2 & \le 6 \\
& 9x_1 + 5x_2 & \le 45 \\
& x_1, x_2 & \ge 0
\end{array}$$

using standard simplex

the optimum dictionary is

$$\begin{array}{rcl} x_1 = & \frac{15}{4} & +\frac{5}{4}\,s_1 - \frac{1}{4}\,s_2 \\ x_2 = & \frac{9}{4} & -\frac{9}{4}\,s_1 + \frac{1}{4}\,s_2 \\ \hline z = & \frac{165}{4} & -\frac{5}{4}\,s_1 - \frac{3}{4}\,s_2 \end{array}$$

adding a new constraint $3x_1 + 2x_2 \le 15$ gives the LP of DS Example
solve it by adding the corresponding constraint $3x_1 + 2x_2 + s_3 = 15$ to the above dictionary
giving the initial (dual feasible) dictionary of DS Example
which is solved in 1 dual simplex pivot

adding a constraint is the basic operation in *cutting plane methods* for integer programming (Handout#37)

Arbitrary Changes

consider a new system still "close to" the original,

maximize $\widetilde{\mathbf{c}}\mathbf{x}$ subject to $\widetilde{\mathbf{A}}\mathbf{x} = \widetilde{\mathbf{b}}$, $\widetilde{\ell} \le \mathbf{x} \le \widetilde{\mathbf{u}}$

assume $\mathbf{B}$ doesn't change (handle such changes by new variables, see Chvátal pp.161–2)

solve the new problem in 2 simplex runs, as follows

1. run primal simplex algorithm
   initial bfs $\overline{\mathbf{x}}$:
   set nonbasic $\overline{x}_j$ to a finite bound $\widetilde{\ell}_j$ or $\widetilde{u}_j$, or to x_j^* if free
   define basic $\overline{x}_j$ by $(*)$
   to make $\overline{\mathbf{x}}$ primal feasible, redefine violated bounds $\widetilde{\ell}, \widetilde{\mathbf{u}}$:
   if j is basic and $\overline{x}_j > \widetilde{u}_j$, new upper bound $= \overline{x}_j$
   if j is basic and $\overline{x}_j < \widetilde{\ell}_j$, new lower bound $= \overline{x}_j$

solve this LP using primal simplex, starting from $\mathbf{B}$, $\overline{\mathbf{x}}$

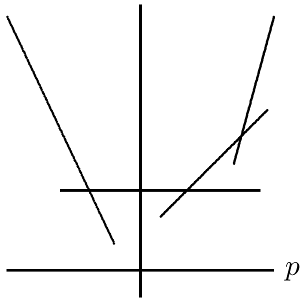
2. run dual simplex algorithm
   change the modified bounds to their proper values $\widetilde{\ell}, \widetilde{\mathbf{u}}$

   in this change no finite bound becomes infinite
   hence it can be handled as in bound changes, using dual simplex algorithm

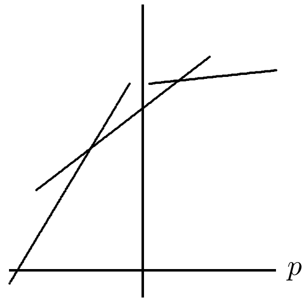
we expect a small # of iterations in both runs an *affine function of* p has the form $ap + b$

given an LP $\mathcal{L}(p)$ with coefficients ($\mathbf{A}$, $\mathbf{b}$, $\mathbf{c}$) affine functions of a parameter p
we wish to analyze the LP as a function of p
  p may be time, interest rate, etc.

Basic Fact. Consider affine functions $f_i(p)$, $i = 1, \ldots, k$.
  $\max\{f_i(p) : i = 1, \ldots, k\}$ is a piecewise-linear concave up function of p.
  $\min\{f_i(p) : i = 1, \ldots, k\}$ is a piecewise-linear concave down function of p.



piecewise linear concave up

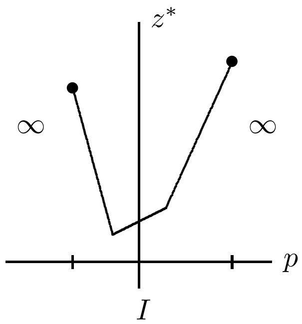


piecewise linear concave down

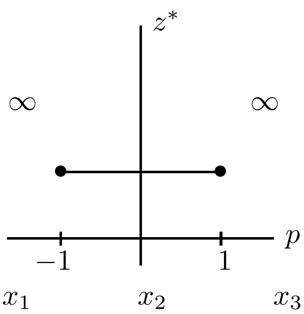
Parametric Costs

$\mathcal{L}(p)$ has the form maximize $(\mathbf{c} + p\mathbf{c}')\mathbf{x}$ subject to $\mathbf{Ax} = \mathbf{b}$, $\ell \le \mathbf{x} \le \mathbf{u}$

Theorem. *For some closed interval I, $\mathcal{L}(p)$ is bounded $\iff p \in I$;*
in I the optimum objective value z^* is a piecewise-linear concave up function of p.



Example.
maximize $(p+1)x_1 + x_2 + (p-1)x_3$ subject to $x_1 \le 0$, $0 \le x_2 \le 1$, $0 \le x_3$



Proof.
wlog $\mathbf{A}$ has full row rank
a basic feasible direction $\mathbf{w}$ has $(\mathbf{c}+p\mathbf{c}')\mathbf{w} > 0$ in some interval
$(-\infty, r)$, (ℓ, ∞), $\mathbf{R}$ or $\emptyset$
thus $\mathcal{L}(p)$ is unbounded in ≤ 2 maximal intervals of the above form
& is bounded in a closed interval I ($I = [\ell, r]$, $(-\infty, r]$, $[\ell, \infty)$, $\mathbf{R}$ or $\emptyset$)

for the 2nd part note that $\mathcal{L}(p)$ has a finite number of normal bfs's
each one $\mathbf{x}$ has objective value $(\mathbf{c}+p\mathbf{c}')\overline{\mathbf{x}}$, an affine function of p □

a basis $\mathbf{B}$ and bfs $\mathbf{x}$ has an interval of optimality, consisting of all values p where
$\mathbf{y} = (\mathbf{c}_B + p\mathbf{c}'_B)\mathbf{B}^{-1}$ is dual feasible
dual feasiblity corresponds to a system of inequalities in p, one per nonbasic variable
hence the interval of optimality is closed

Algorithm to Find I and z^* ("walk the curve")

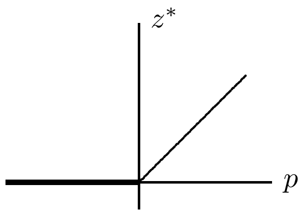
solve $\mathcal{L}(p)$ for some arbitrary p, using the simplex algorithm
let $\mathbf{B}$ be the optimum basis, with interval of optimality $[\ell, r]$
if $\mathcal{L}(p)$ is unbounded the algorithm is similar

at r, $\mathbf{B}$ is dual feasible & ≥ 1 of the corresponding inequalities holds with equality
increasing r slightly, these tight inequalities determine the entering variable in a simplex pivot
do this simplex pivot to find a basis $\mathbf{B}'$ with interval of optimality $[r, r']$
1 pivot often suffices but more may be required

continue in the same way to find the optimal bases to the right of r
stop when a basis has interval $[r'', \infty)$, perhaps with unbounded objective

similarly find the optimal bases to the left of ℓ

Example.
maximize px_1 subject to $x_1 + x_2 = 1$, $x_1, x_2 \ge 0$



optimum basis: x_2 x_1

Parametric R.H. Sides

$\mathcal{L}(p)$ has form maximize $\mathbf{cx}$ subject to $\mathbf{Ax} = \mathbf{b} + p\mathbf{b}'$, $\ell + p\ell' \le \mathbf{x} \le \mathbf{u} + p\mathbf{u}'$

assume
$(*)$ some $\mathcal{L}(p)$ has an optimum solution

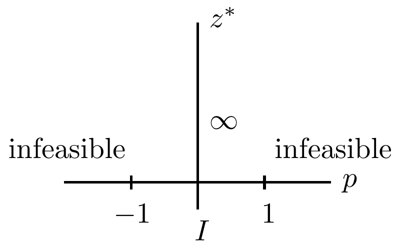
$(*)$ implies no $\mathcal{L}(p)$ is unbounded
since some dual $\mathcal{L}^*(p_0)$ has an optimum $\Longrightarrow$ every $\mathcal{L}^*(p)$ is feasible

Theorem. *Assuming $(*)$ there is a closed interval $I \ni$ ($\mathcal{L}(p)$ is feasible $\Longleftrightarrow p \in I$); in I the optimum objective value exists & is a piecewise-linear concave down function of p.*

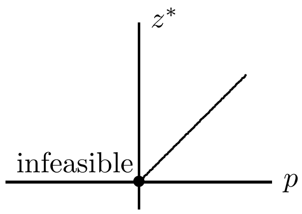
Proof. by duality $\square$

if $(*)$ fails, any $\mathcal{L}(p)$ is infeasible or unbounded

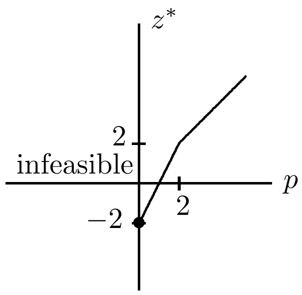
Examples.
1. maximize x_1 subject to $x_2 = p$, $-1 \le x_2 \le 1$



2. maximize x_1 subject to $x_1 + x_2 = p$, $x_1, x_2 \ge 0$



3. maximize x_1 subject to $x_1 + x_2 = 2p - 2$, $-2 \le x_1 \le p$, $x_2 \ge 0$



the cutting plane method for ILP starts with the LP relaxation,
and repeatedly adds a new constraint
the new constraint eliminates some nonintegral points from the relaxation's feasible region
eventually the LP optimum is the ILP optimum

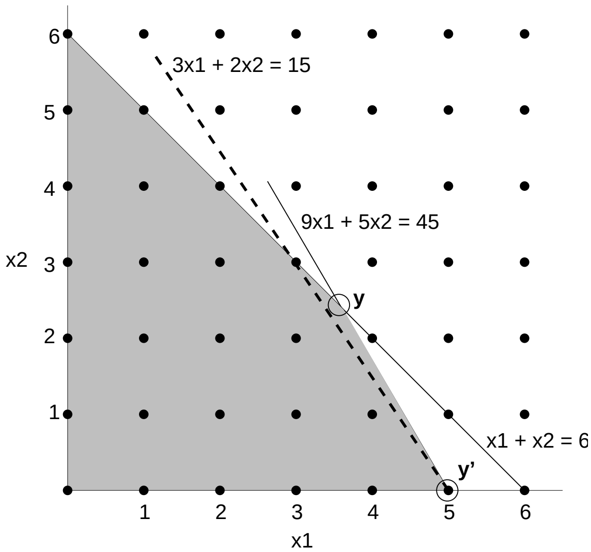
DS Example (Handout#34) cont'd.

ILP:

$$\begin{array}{llcl} \text{maximize } z = 8x_1+5x_2 & & & \\ \text{subject to} & x_1+\ x_2 & \le & 6 \\ & 9x_1+5x_2 & \le & 45 \\ & x_1, x_2 & \in & \mathbf{Z}^+ \end{array}$$

LP with cutting plane:

$$\begin{array}{llcl} \text{maximize } z = 8x_1+5x_2 & & & \\ \text{subject to} & x_1+\ x_2 & \le & 6 \\ & 9x_1+5x_2 & \le & 45 \\ & 3x_1+2x_2 & \le & 15 \\ & x_1, x_2 & \ge & 0 \end{array}$$



The cutting plane $3x_1 + 2x_2 = 15$ moves the LP optimum from $\mathbf{y} = (15/4, 9/4)$ to the ILP optimum $\mathbf{y}' = (5, 10)$. (Fig. from WV).

Method of Fractional Cutting Planes (Gomory, '58)

consider an ILP: maximize $\mathbf{cx}$ subject to $\mathbf{Ax} = \mathbf{b}$, $\mathbf{x}$ integral
we allow all coefficients $\mathbf{A}$, $\mathbf{b}$, $\mathbf{c}$ to be real-valued,
although they're usually integral in the given problem

suppose the LP relaxation has a fractional optimum $\mathbf{x}$

in the optimum dictionary consider the equation for a basic variable x_i whose value b_i is fractional:

(∗) $$x_i = b_i - \sum_{j\in N} a_{ij}x_j$$

let f_i denote the fractional part of b_i, i.e., $f_i = b_i - \lfloor b_i \rfloor$
similarly let f_{ij} denote the fractional part of a_{ij}

in the optimum ILP solution, the r.h.s. of (∗) is an integer
it remains integral even if we discard integral terms
$\therefore\ f_i - \sum_{j\in N} f_{ij}x_j$ is an integer, say a
$a \le f_i < 1 \Longrightarrow a \le 0$
thus any integral solution satisfies

(†) $$f_i - \sum_{j\in N} f_{ij}x_j \le 0$$

with integral slack

the current LP optimum doesn't satisfy (†) (since all nonbasic x_j are 0)
so adding (†) to the constraints, with an integral slack variable, gives an equivalent ILP
with a new LP optimum

DS Example (Handout#34) cont'd.
the optimum dictionary of (Handout#35)

$$\begin{array}{rcrl}
x_1 &=& \frac{15}{4} & +\frac{5}{4}s_1 - \frac{1}{4}s_2 \\
x_2 &=& \frac{9}{4} & -\frac{9}{4}s_1 + \frac{1}{4}s_2 \\
\hline
z &=& \frac{165}{4} & -\frac{5}{4}s_1 - \frac{3}{4}s_2
\end{array}$$

has x_1 nonintegral
x_1's equation is $x_1 = \frac{15}{4} - (-\frac{5}{4})s_1 - \frac{1}{4}s_2$
keeping only fractional parts the r.h.s. is $\frac{3}{4} - \frac{3}{4}s_1 - \frac{1}{4}s_2$
so the cutting plane is $\frac{3}{4} - \frac{3}{4}s_1 - \frac{1}{4}s_2 \le 0$
equivalently $3 \le 3s_1 + s_2$, or in terms of original variables, $12x_1 + 8x_2 \le 60$, $3x_1 + 2x_2 \le 15$

Summary of the Algorithm

solve the LP relaxation of the given IP
while the solution is fractional **do**
add a cut (†) to the LP
resolve the new LP
/∗ use the dual simplex algorithm, since we're just adding a new constraint ∗/

Example. DS Example in Handouts#34–35 show how the ILP of p.1 is solved

Gomory proved this algorithm solves the ILP in a finite number of steps

Refinements:
choosing an f_i close to half is recommended in practice
can discard cuts that become inactive

in practice the method can be slow – more sophisticated cutting strategies are used

Remarks

1. if the given ILP has constraints $\mathbf{Ax} \le \mathbf{b}$ rather than equalities,
   we require $\mathbf{A}$ & $\mathbf{b}$ both integral, so all slack variables are integral
   if this doesn't hold, can scale up $\mathbf{A}$ & $\mathbf{b}$

2. the fractional cutting plane method can be extended to mixed integer programs (MIP)

3. cutting planes can be used within a branch-and-bound algorithm
   to strengthen the bound on the objective function

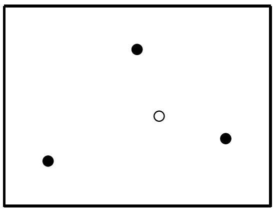
we give some geometric consequences of the characterization of Handout #32
for inconsistent sytems of inequalities:

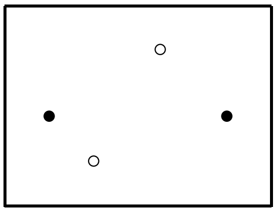
Corollary. $\mathbf{Ax} \le \mathbf{b}$ *is infeasible* $\iff$ *some subsystem of* $\le n+1$ *inequalities is infeasible.*

say a hyperplane $\mathbf{ax} = b$ *strictly separates* sets $R, G \subseteq \mathbf{R}^n$ if
each $\mathbf{r} \in R$ has $\mathbf{ar} > b$ & each $\mathbf{g} \in G$ has $\mathbf{ag} < b$

Theorem. *Consider a finite set of points of* $\mathbf{R}^n$*, each one colored red or green.*
Some hyperplane strictly separates the red & green points $\iff$
this holds for every subset of $n+2$ *points.*

Example.
red & green points in the plane, can be separated by a line unless
there are 4 points in 1 of these 2 configurations:





Proof.
a set of red & green points can be strictly separated $\iff$
some hyperplane $\mathbf{ax} = b$ has $\mathbf{ar} \le b$ & $\mathbf{ag} \ge b+1$ for each red point $\mathbf{r}$ & each green point $\mathbf{g}$
(by rescaling)

thus our given set can be separated $\iff$
this system of inequalities is feasible for unknowns $\mathbf{a}$ & b:
$\mathbf{ar} \le b$ for each given red point $\mathbf{r}$
$\mathbf{ag} \ge b+1$ for each given green point $\mathbf{g}$

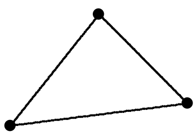
since there are $n+1$ unknowns, the Corollary gives the Theorem $\square$

a subset of $\mathbf{R}^n$ is *convex* if any 2 of its points can "see" each other–
$\mathbf{x}, \mathbf{y} \in C \Longrightarrow$ the line segment between $\mathbf{x}$ & $\mathbf{y}$ is contained in C

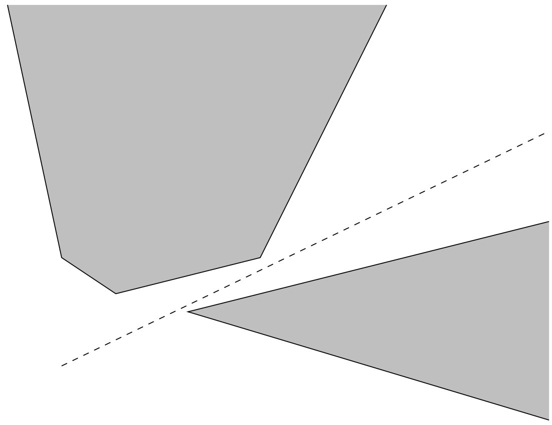
a similar use of the Corollary, plus some facts on convex sets,
implies this famous result (Chvátal p.266):

Helly's Theorem. *Consider a finite collection of* $\ge n+1$ *convex sets in* $\mathbf{R}^n$*.*
They have a common point if every $n+1$ *sets do.*

Helly's Theorem can't be improved to n sets, e.g., take 3 lines the plane:



we can also separate two polyhedra, e.g.,



Separation Theorem for Polyhedra.
Two disjoint convex polyhedra in $\mathbf{R}^n$ *can be strictly separated by a hyperplane.*

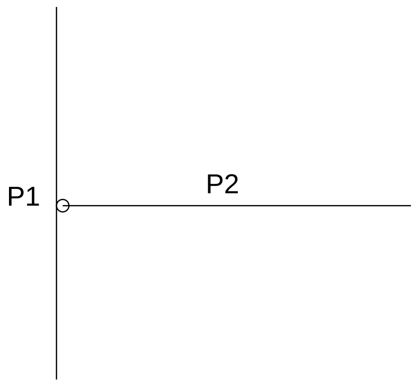
Proof.
let the 2 polyhedra be P_i, $i = 1, 2$
corresponding to systems $\mathbf{A}_i\mathbf{x} \leq \mathbf{b}_i$, $i = 1, 2$
assume both P_i are nonempty else the theorem is trivial

disjointness $\Longrightarrow$ no point satisfies both systems
$\Longrightarrow$ there are vectors $\mathbf{y}_i \geq \mathbf{0}$ satisfying
$\mathbf{y}_1\mathbf{A}_1 + \mathbf{y}_2\mathbf{A}_2 = \mathbf{0}$, $\mathbf{y}_1\mathbf{b}_1 + \mathbf{y}_2\mathbf{b}_2 < 0$

set $\mathbf{y}_1\mathbf{A}_1 = \mathbf{h}$, so $\mathbf{y}_2\mathbf{A}_2 = -\mathbf{h}$
$\mathbf{h}$ is a row vector

for $\mathbf{x} \in P_1$, $\mathbf{hx} = \mathbf{y}_1\mathbf{A}_1\mathbf{x} \leq \mathbf{y}_1\mathbf{b}_1$
for $\mathbf{x} \in P_2$, $\mathbf{hx} = -\mathbf{y}_2\mathbf{A}_2\mathbf{x} \geq -\mathbf{y}_2\mathbf{b}_2$

since $\mathbf{y}_1\mathbf{b}_1 < -\mathbf{y}_2\mathbf{b}_2$, taking c as a value in between these 2 gives
$\mathbf{hx} = c$ a hyperplane strictly separating P_1 & P_2 $\square$



P_1 is a closed line segment, hence a convex polyhderon.
P_2 is a half-open line segment – its missing endpoint is in P_1.
P_1 & P_2 cannot be separated.

Remarks

1. the assumption P_i nonempty in the above argument ensures $\mathbf{h} \neq \mathbf{0}$
   (since $\mathbf{h} = \mathbf{0}$ doesn't separate any points)

   this argument is a little slicker than Chvátal

2. for both theorems of this handout, Chvátal separates sets A & B using 2 disjoint half-spaces
   i.e., points $\mathbf{x} \in A$ have $\mathbf{hx} \leq c$, points $\mathbf{x} \in B$ have $\mathbf{hx} \geq c + \epsilon$

   for finite sets of points, the 2 ways to separate are equivalent
   but not for infinite sets – e.g., we can strictly separate the sets $\mathbf{hx} > c$ & $\mathbf{hx} < c$
   but not with disjoint half-spaces

   separation by disjoint half-spaces implies strict separation

   so the above Separation Theorem would be stronger if we separated by disjoint half-spaces
   that's what we did in the proof!, so the stronger version is true
   (why not do this in the first place? – simplicity)

this problem is to route specified quantities of homogeneous goods, minimizing the routing cost more precisely:

let G be a directed graph on n vertices and m edges
the *undirected version* of G ignores all edge directions
for simplicity assume it's connected

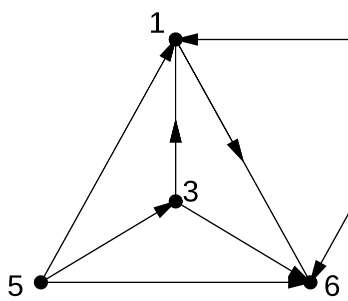


Fig.1. Graph G.

each vertex i has a *demand* b_i, & we call i a $\begin{cases} \textit{sink} & b_i > 0 \\ \textit{source} & b_i < 0 \\ \textit{intermediate (transshipment) node} & b_i = 0 \end{cases}$

for simplicity assume $\sum_i b_i = 0$

each edge ij has a *cost* c_{ij}, the cost of shipping 1 unit of goods from i to j

we want to satisfy all demands exactly, and minimize the cost

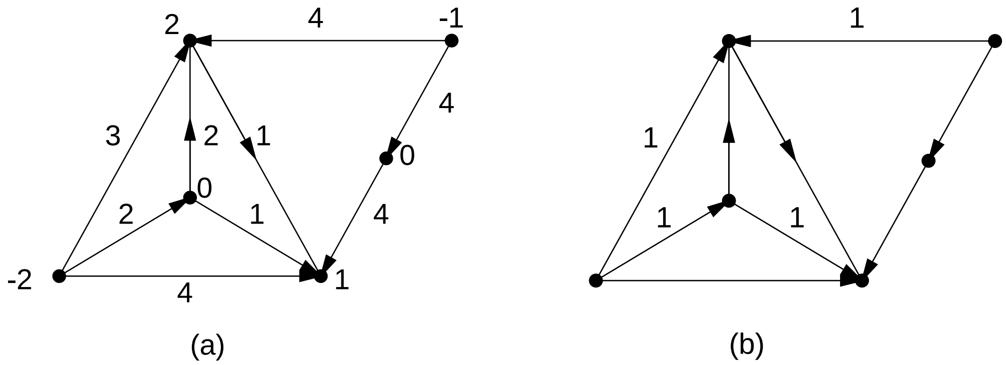


Fig.2. (a) Graph G with vertex demands & edge costs. (b) Optimum transshipment, cost 10. Edge labels give # units shipped on the edge; 0 labels are omitted.
1 unit is shipped along path 5,3,6 – vertex 3 functions as a transshipment node.

Special Cases of the Transshipment Problem.
assignment problem & its generalization, transportation problem
shortest path problem

Exercise. Model the single-source shortest path problem as a transhipment problem.

we state the problem as an LP:

the (*node-arc*) *incidence matrix* of G: $n \times m$ matrix $\mathbf{A}$
  the column for edge ij has ith entry -1, jth entry $+1$ & all others 0
Example. the column for edge $(3,1)$ is $[1\ 0\ \ -1\ 0\ 0\ 0]^T$

edge ij has a variable x_{ij} giving the number of units shipped from i to j

Transshipment Problem: minimize $\mathbf{cx}$
subject to $\mathbf{Ax} = \mathbf{b}$
$\mathbf{x} \geq \mathbf{0}$

it's obvious that any feasible routing satisfies this LP
the following exercise proves that any $\mathbf{x}$ feasible to the LP corresponds to a feasible routing

Exercise. (a) Assume all costs are nonnegative (as one expects). Prove that the following algorithm translates any feasible LP solution $\mathbf{x}$ into a valid routing for the transshipment problem.

> Let P be a path from a source to a sink, containing only edges with positive x_{ij}. Let $\mu = \min\{x_{ij} : ij \in P\}$. Ship μ units along P. Then reduce $\mathbf{b}$ and $\mathbf{x}$ to reflect the shipment, and repeat the process. Stop when there are no sources.

(b) Modify the algorithm so it works even when there are negative costs. How do negative costs change the nature of the problem?

the *Dual Transhipment Problem*: maximize $\mathbf{yb}$
subject to $y_j \leq y_i + c_{ij}$, for each arc ij

the dual variables have a nice economic interpretation as prices:
the dual constraints say
  (price at node i) + (shipment cost to j) $\geq$ (price at node j)

we'll solve the transhipment problem using the minimizing simplex algorithm (Handout#27, p.3)

Exercise. Show that if (y_i) is dual optimal, so is $(y_i + a)$ for any constant a.

Linear Algebra & Graph Theory

the constraints $\mathbf{Ax} = \mathbf{b}$ of the transshipment problem do not have full row rank:
the rows of $\mathbf{A}$ sum to $\mathbf{0}$ since every edge leaves & enters a vertex

form $\widetilde{\mathbf{A}}$ & $\widetilde{\mathbf{b}}$ by discarding the row for vertex r (choose r arbitrarily)
the *reduced system* is the transhipment problem with constraints $\widetilde{\mathbf{A}}\mathbf{x} = \widetilde{\mathbf{b}}$
a solution to the reduced system is a solution to the original problem
the discarded equation holds automatically since the entries of $\mathbf{b}$ sum to 0
now we'll show the reduced system has full row rank

the phrases "spanning tree of G" & "cycle of G" refer to the undirected version of G

when we traverse a cycle, an edge is called *forward* if it's traversed in the correct direction,
else *backward*
Example. in cycle $1, 6, 3$, edge $(6, 3)$ is backward, the others are forward
for edge ij in the cycle, $sign(ij)$ is $+1$ (-1) if ij is forwards (backwards)

Lemma. *A basis of the reduced system is a spanning tree of G.*

Example. the solution of Handout#39, Fig.2(b) is not a spanning tree, but we can enlarge it to a spanning tree with edges shipping 0 units:

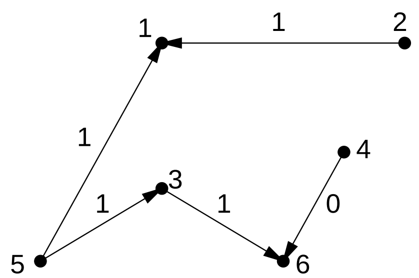


Fig.3. Edges forming an optimum (degenerate) basis.

Proof.

Claim 1. The edges of a cycle have linearly dependent columns, in both $\mathbf{A}$ & $\widetilde{\mathbf{A}}$
Proof.
it suffices to prove it for $\mathbf{A}$

traverse the cycle, adding the edge's column times its sign
the sum is 0 $\diamondsuit$

Example. for the cycle $1, 6, 3$ & $\mathbf{A}$ we get $[-1\ 0\ 0\ 0\ 0\ 1]^T - [0\ 0\ \ -1\ 0\ 0\ 1]^T + [1\ 0\ \ -1\ 0\ 0\ 0]^T = \mathbf{0}$

Claim 2. The columns of a forest are linearly independent, in $\mathbf{A}$ & $\widetilde{\mathbf{A}}$

Proof.

it suffices to prove this for $\widetilde{\mathbf{A}}$

suppose a linear combination of the edges sums to $\mathbf{0}$
an edge incident to a leaf $\neq r$ has coefficient 0
repeat this argument to eventually show all coefficients are 0 $\diamondsuit$

the lemma follows
since a basis of the reduced system consists of $n-1$ linearly independent columns of $\widetilde{\mathbf{A}}$ $\square$

Exercise 1. (a) Show a basis (spanning tree) has a corresponding matrix $\mathbf{B}$ in $\widetilde{\mathbf{A}}$ whose rows & columns can be ordered so the matrix is upper triangular, with ± 1's on the diagonal.

In (b)–(c), root the spanning tree at r. (b) The equation $\mathbf{Bx} = \mathbf{b}$ gives the values of the basic variables. Show how to compute $\mathbf{x}$ by traversing T bottom-up. (c) The equation $\mathbf{yB} = \mathbf{c}$ gives the values of the duals. Show how to compute $\mathbf{y}$ by traversing T top-down.

Execute your algorithms on Fig.5(a).

the algorithms of (b) & (c) use only addition and subtraction, no $\times$ or $/$
the Fundamental Theorem shows some basis $\mathbf{B}$ is optimum. so we get (see also Handout#61):

Integrality Theorem. *If* $\mathbf{b}$ *is integral, the transshipment problem has an optimum integral solution* $\mathbf{x}$ *(regardless of* $\mathbf{c}$*).*

Pivotting

how do we pivot an edge into the current basis?

let T be the current basis; to add edge ij to the basis:
$T + ij$ contains a cycle C
traverse C, starting out by going from i to j
suppose we increase each x_{uv}, $uv \in C$, by $sign(uv) \cdot t$ ($t \geq 0$)
x_{ij} increases, as desired
the quantity $\widetilde{\mathbf{A}}\mathbf{x}$ doesn't change, since at each vertex u, 2 edges change and the changes balance

so $\mathbf{x}$ remains feasible, as long as it stays nonnegative
take $t = \min\{x_{uv} : uv \text{ a backwards edge}\}$
this is the largest t possible; some backwards edge drops to 0 and is the leaving variable

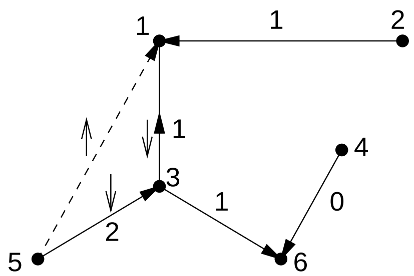


Fig.4. A (suboptimal) basis. Pivotting edge $(5, 1)$ into the basis gives Fig.3.

Prices

each vertex maintains a dual variable (it's "price") defined by $y_r = 0$ and $\mathbf{yB} = \mathbf{c}$ (Exercise 1(c))

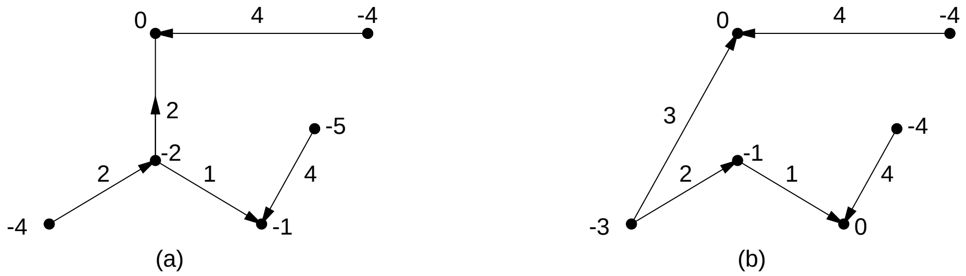


Fig.5. Prices for the bases of Fig.4 and 3 respectively. r is the top left vertex (as in Fig.1, Handout#39). Each basic edge ij satisfies $y_i + c_{ij} = y_j$. (b) gives optimum prices: $\sum y_i b_i = 10$.

Note: y_r doesn't exist in the reduced system
but the constraints for edges incident to r are equivalent to
usual constraints $y_i + c_{ij} \geq y_j$ with $y_r = 0$

Network Simplex Algorithm
this algorithm implements the (basic) simplex algorithm for the transshipment problem

each iteration starts with a basis $\mathbf{B}$ (spanning tree T) and bfs $\mathbf{x}$

Entering Variable Step
Solve $\mathbf{yB} = \mathbf{c}_B$ by traversing T top-down (Exercise 1(c)).
Choose any (nonbasic) edge $ij \ni c_{ij} < y_j - y_i$. /* underbidding */
If none exists, stop, B is an optimum basis.

Leaving Variable Step
Execute a pivot step by traversing edge ij's cycle C and finding t.
If $t = \infty$, stop, the problem is unbounded. /* impossible if $\mathbf{c} \geq \mathbf{0}$ */
Otherwise choose a backwards edge uv that defines t.

Pivot Step
Update $\mathbf{x}$: change values along C by $\pm t$.
In T replace edge uv by ij. $\square$

Example. in Fig.5(a) edge $(5, 1)$ can enter, since $3 < 0 - (-4)$
the pivot step (Fig.4) gives Fig.5(b), optimum.

this code involves additions and subtractions, no $\times$ or / (as expected!)

like simplex,
the network simplex algorithm is very fast in practice
although no polynomial time bound is known (for any pivotting rule)

the primal-dual method (Chvátal Ch.23) leads to polynomial algorithms for polynomial-time algorithms, we assume the given data $\mathbf{A}$, $\mathbf{b}$, $\mathbf{c}$ is integral

the ellipsoid algorithm solves the problem of *Linear Strict Inequalities (LSI)*:

Given: "open polyhedron" P defined by $\mathbf{Ax} < \mathbf{b}$
as usual this means m strict inequalities in n unknowns $\mathbf{x}$

Task: find some $\mathbf{x} \in P$ or declare $P = \emptyset$

note $P \subseteq \mathbf{R}^n$; our entire discussion takes place in $\mathbf{R}^n$

recall LI (Linear Inequalities) is equivalent to LP (Exercise of Handout#18)
we can solve an LI problem using an algorithm for LSI (using integrality of $\mathbf{A}$, $\mathbf{b}$)

define L = size of input, i.e., (total # bits in $\mathbf{A}$, $\mathbf{b}$) (see Handout#69)

Why Strict Inequalities?
using integrality we can prove
$P \neq \emptyset \implies \text{volume}(P) \geq 2^{-(n+2)L}$

Proof Sketch:
a *simplex* in $\mathbf{R}^n$ is the convex hull of $n+1$ vertices
any polytope P decomposes into a finite number of simplices
each simplex has vertices that are cornerpoints of P

$P \neq \emptyset \implies \text{interior}(P) \neq \emptyset$
$\implies P$ contains a simplex of positive volume, with integral cornerpoints
the volume bound follows from the integrality of $\mathbf{A}$ & $\mathbf{b}$ and the Exercise of Handout#25

Ellipsoids (see Fig.1)
an *ellipsoid* is the image of the unit sphere under an affine transformation, i.e.,
an ellipsoid is $\{\mathbf{c} + \mathbf{Qx} : \|\mathbf{x}\| \leq 1\}$ for an $n \times n$ nonsingular matrix $\mathbf{Q}$

equivalently an ellipsoid is $\{\mathbf{x} : (\mathbf{x}-\mathbf{c})^T\mathbf{B}^{-1}(\mathbf{x}-\mathbf{c}) \leq 1\}$, for a positive definite matrix $\mathbf{B}$

Proof. (matrix background in Handouts#65,64)
the ellipsoid is the set of points $\mathbf{y}$ with $\|\mathbf{Q}^{-1}(\mathbf{y}-\mathbf{c})\| \leq 1$
i.e., $(\mathbf{y}-\mathbf{c})^T(\mathbf{Q}^{-1})^T\mathbf{Q}^{-1}(\mathbf{y}-\mathbf{c}) \leq 1$
so $\mathbf{B} = \mathbf{QQ}^T$, $\mathbf{B}$ is positive definite $\square$

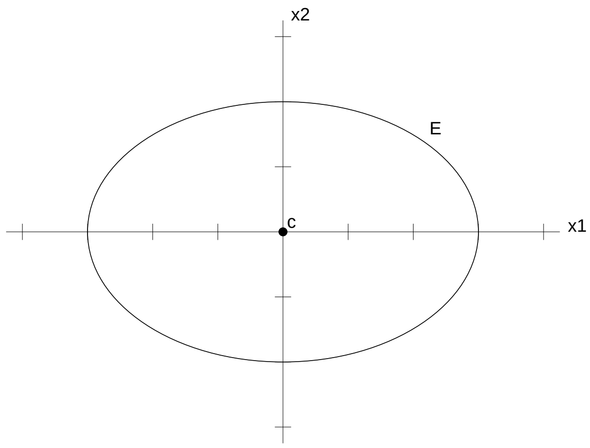


Fig.1. Ellipse $\dfrac{x_1^2}{9} + \dfrac{x_2^2}{4} = 1$; equivalently center $\mathbf{c} = \mathbf{0}$, $\mathbf{B} = \begin{bmatrix} 9 & 0 \\ 0 & 4 \end{bmatrix}$.

High-level Algorithm
construct a sequence of ellipsoids E_r, $r = 0, \ldots, s$
each containing P
with volume shrinking by a factor $2^{-1/2(n+1)}$
stop when either
(i) the center of E_s is in P, or
(ii) volume$(E_s) < 2^{-(n+2)L}$ ($\Longrightarrow P = \emptyset$)

Initialization and Efficiency
we use a stronger version of the basic volume fact:
$P \neq \emptyset \Longrightarrow$ volume$(P \cap \{\mathbf{x} : |x_j| < 2^L,\ j = 1, \ldots, n\}) \geq 2^{-(n+2)L}$

thus E_0 can be a sphere of radius $n2^L$, center $\mathbf{0}$

\# iterations $= O(n^2 L)$
more precisely suppose we do $N = 16n(n+1)L$ iterations without finding a feasible point

our choice of E_0 restricts all coordinates to be $\leq n2^L$
i.e., E_0 is contained in a box with each side $2n2^L \leq 2^{2L}$
$\Longrightarrow$ volume$(E_0) \leq 2^{2nL}$

after N iterations the volume has shrunk by a factor $2^{-N/2(n+1)} = 2^{-8nL}$

$\therefore$ after N iterations the ellipse has volume $\leq 2^{2nL - 8nL} \leq 2^{-6nL} < 2^{-(n+2)L}$
$\Longrightarrow P = \emptyset$

Implementing the High-level Algorithm
if $P \subseteq E$, & center $\mathbf{c}$ of E is not in P,
there is a violated hyperplane, i.e., $\mathbf{A}_{i\cdot}\mathbf{c} \ge b_i$

for the corresponding half-space H (i.e., $\mathbf{A}_{i\cdot}\mathbf{x} < b_i$)
$P \subseteq E \cap H$
the algorithm replaces E by a smaller ellipsoid that contains $E \cap H$, given by the following theorem

Ellipsoid Shrinking Theorem.
For an ellipsoid E, let H be a half-space containing the center.
$\exists$ *an ellipsoid E' containing $E \cap H$ with* $\text{volume}(E') \le 2^{-1/2(n+1)} \times \text{volume}(E)$. $\square$

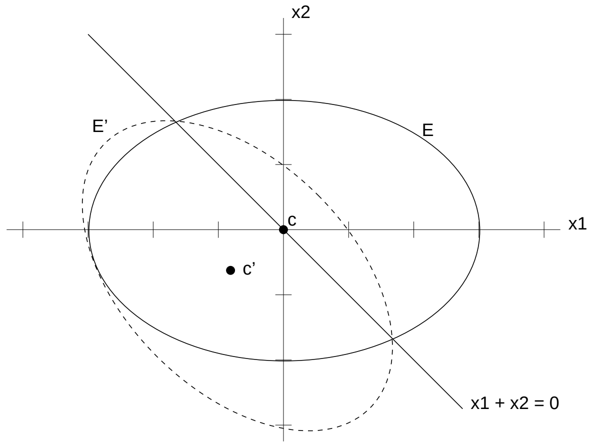


Fig.2. E', with center $\mathbf{c}' = (-3/\sqrt{13}, -4/3\sqrt{13})^T$, $\mathbf{B} = \begin{bmatrix} 84/13 & -32/13 \\ -32/13 & 496/117 \end{bmatrix}$
contains intersection of E of Fig.1 & half-space $x_1 + x_2 \le 0$

Ellipsoid Algorithm
Initialization
Set $N = 1 + 16n(n+1)L$.
Set $\mathbf{p} = \mathbf{0}$ and $\mathbf{B} = n^2 2^{2L}\mathbf{I}$.
/* The ellipsoid is always $\{\mathbf{x} : (\mathbf{x} - \mathbf{p})^T \mathbf{B}^{-1}(\mathbf{x} - \mathbf{p}) \le 1\}$.
The initial ellipse is a sphere centered at $\mathbf{0}$ of radius $n2^L$. */

Main Loop
Repeat the Shrink Step N times (unless it returns).
If it never returns, return “infeasible”.

Shrink Step
If $\mathbf{Ap} < \mathbf{b}$ then return $\mathbf{p}$ as a feasible point.
Choose a violated constraint, i.e., an i with $\mathbf{A}_{i\cdot}\mathbf{p} \ge \mathbf{b}_i$.
Let $\mathbf{a} = \mathbf{A}_{i\cdot}^T$.
Let $\mathbf{p} = \mathbf{p} - \dfrac{1}{n+1}\dfrac{\mathbf{Ba}}{\sqrt{\mathbf{a}^T\mathbf{Ba}}}$.
Let $\mathbf{B} = \dfrac{n^2}{n^2-1}\left(\mathbf{B} - \dfrac{2}{n+1}\dfrac{(\mathbf{Ba})(\mathbf{Ba})^T}{\mathbf{a}^T\mathbf{Ba}}\right)$.

Remarks

1. the ellipsoids of the algorithm must be approximated,
   since their defining equations involve square roots

   this leads to a polynomial time algorithm

2. but the ellipsoid algorithm doesn't take advantage of sparsity

3. it can be used to get polynomial algorithms for LPs with exponential #s of constraints!
   e.g., the Held-Karp TSP relaxation (Handout#1)

   note the derivation of N is essentially independent of m

   to execute ellipsoid on an LP, we only need an efficient algorithm for
   the separation problem:
   given $\mathbf{x}$, decide if $\mathbf{x} \in P$
   if $\mathbf{x} \notin P$, find a violated constraint

Convex Programming

let C be a convex set in $\mathbf{R}^n$
i.e., $\mathbf{x}, \mathbf{y} \in C \Longrightarrow \theta\mathbf{x} + (1-\theta)\mathbf{y} \in C$ for any $\theta \in [0,1]$

Problem: min $\mathbf{cx}$ s.t. $\mathbf{x} \in C$

Fundamental Properties for Optimization

1. for our problem a local optimum is a global optimum

   Proof.
   let $\mathbf{x}$ be a local optimum & take any $\mathbf{y} \in C$
   take $\theta \in (0,1]$ small enough so $\mathbf{c}\big((1-\theta)\mathbf{x} + \theta\mathbf{y}\big) \geq \mathbf{cx}$
   thus $(1-\theta)\mathbf{cx} + \theta\mathbf{cy} \geq \mathbf{cx}, \quad \mathbf{cy} \geq \mathbf{cx}$ $\square$

2. $\mathbf{x} \notin C \Longrightarrow \exists$ a separating hyperplane, i.e.,
   $\mathbf{by} > a$ for all $\mathbf{y} \in C$ and $\mathbf{bx} < a$

   proved in Handout#38

because of these properties the ellipsoid algorithm solves our convex optimization problem,
assuming we can recognize points in C & solve the separation problem

a prime example is semidefinite programming:

in the following $\mathbf{X}$ denotes a square matrix of variables
and $\widehat{\mathbf{X}}$ denotes the column vector of $\mathbf{X}$'s entries

the *semidefinite programming problem* is this generalization of LP:

$$\begin{array}{lll} \text{maximize } z = & \mathbf{c}\widehat{\mathbf{X}} & \\ \text{subject to} & \mathbf{A}\widehat{\mathbf{X}} & \le \mathbf{b} \\ & \mathbf{X} & \text{an } n \times n \text{ symmetric positive semidefinite matrix} \end{array}$$

Example. min x s.t. $\begin{bmatrix} x & -1 \\ -1 & 1 \end{bmatrix}$ is PSD
this problem is equivalent to min x s.t. $xv^2 - 2vw + w^2 \ge 0$ for all v, w
$x = 1$ is the optimum
(taking $v = w = 1$ shows $x \ge 1$, & clearly $x = 1$ is feasible)

in general, the feasible region is convex
a convex combination of PSD matrices is PSD

the separation problem is solved by finding $\mathbf{X}$'s eigenvalues
$\mathbf{X}$ not PSD $\Longrightarrow$ it has a negative eigenvalue λ
let $\mathbf{v}$ be the corresponding eigenvector
$\mathbf{v}^T X \mathbf{v} = \mathbf{v}^T \lambda \mathbf{v} < 0$
so $\mathbf{v}^T X \mathbf{v} \ge 0$ is a "separating hyperplane"
i.e., it separates the current solution from the feasible region
& can be used to construct the next ellipse

Conclusion: *For any $\epsilon > 0$, any semidefinite program can be solved by the ellipsoid algorithm to within an additive error of ϵ, in time polynomial in the input size and* $\log(1/\epsilon)$.

Examples:
1. $\theta(G)$ (Handout#50) is computed in polynomial time using semidefinite programming

2. .878 approximation algorithms for MAX CUT & MAX 2 SAT
(Goemans & Williamson, *STOC '94*); see Handout#44

Remarks.

1. SDP includes convex QP as a special case (Exercise below)

2. SDP also has many applications in control theory

3. work on SDP started in the 80's
   interior point methods (descendants of Karmarkar) run in polynomial time
   & are efficient in practice, especially on bigger problems

Exercise. We show SDP includes QP (Handout#42) and more generally, convex quadratically constrainted quadratic programming (QCQP).

(*i*) For $\mathbf{x} \in \mathbf{R}^n$, show the inequality

$$(\mathbf{Ax}+\mathbf{b})^T(\mathbf{Ax}+\mathbf{b}) \le \mathbf{cx}+\mathbf{d}$$

is equivalent to

$$\begin{bmatrix} \mathbf{I} & \mathbf{Ax}+\mathbf{b} \\ (\mathbf{Ax}+\mathbf{b})^T & \mathbf{cx}+\mathbf{d} \end{bmatrix} \text{ is PSD}$$

.

Hint. Just use the definition of PSD. Recall ax^2+bx+c is always nonnegative iff $b^2 \le 4ac$ and $a+c \ge 0$.

(*ii*) Show QP is a special case of SDP.

(*iii*) Show QCQP is a special case of SDP. QCQP is minimizing a quadratic form (as in QP) subject to quadratic constraints

$$(\mathbf{Ax}+\mathbf{b})^T(\mathbf{Ax}+\mathbf{b}) \le \mathbf{cx}+\mathbf{d}.$$

a *Quadratic Program (QP)* $\mathcal{Q}$ has the form

$$\begin{aligned} &\text{minimize } \tfrac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{c}\mathbf{x} \\ &\text{subject to } \mathbf{A}\mathbf{x} \ge \mathbf{b} \\ &\qquad\qquad\quad \mathbf{x} \ge \mathbf{0} \end{aligned}$$

$\mathbf{Q}$ is an $n \times n$ symmetric matrix (*wlog*)
we have a *convex quadratic program* if $\mathbf{Q}$ is positive semi-definite, i.e., $\mathbf{x}^T\mathbf{Q}\mathbf{x} \ge 0$ for every $\mathbf{x}$
justification: the objective function is convex, i.e., concave up, $\Longleftrightarrow$ $\mathbf{Q}$ is PSD

Exercises.

1. Prove the above, i.e., denoting the objective function as $c(\mathbf{x})$, we have
for all $\mathbf{x}$, $\mathbf{y}$, & θ with $0 \le \theta \le 1$, $c(\theta\mathbf{x} + (1-\theta)\mathbf{y}) \le \theta c(\mathbf{x}) + (1-\theta)c(\mathbf{y}) \iff \mathbf{Q}$ is PSD.
(*Hint.* Just the definition of PSD is needed.)

2. Show the objective of any convex QP can be written as

$$\text{minimize } (\mathbf{D}\mathbf{x})^T(\mathbf{D}\mathbf{x}) + \mathbf{c}\mathbf{x}$$

for $\mathbf{D}$ an $n \times n$ matrix. And conversely, any such objective gives a convex QP. (*Hint.* Recall Handout#64.) So again the restriction to PSD $\mathbf{Q}$'s is natural.

Example 1. Let P be a convex polyhedron & let $\mathbf{p}$ be a point not in P. Find the point in P closest to $\mathbf{p}$.

we want to minimize $(\mathbf{x}-\mathbf{p})^T(\mathbf{x}-\mathbf{p}) = \mathbf{x}^T\mathbf{x} - 2\mathbf{x}^T\mathbf{p} + \mathbf{p}^T\mathbf{p}$
so we have a QP with $\mathbf{Q} = \mathbf{I}$, $\mathbf{c} = -\mathbf{p}^T$
$\mathbf{Q}$ is PD

Example 2. Let P & P' be 2 convex polyhedra that do not intersect. Find the points of P & P' that are closest together.

we want to minimize $(\mathbf{x}-\mathbf{y})^T(\mathbf{x}-\mathbf{y})$
this gives a QP with $\mathbf{c} = \mathbf{0}$, & $\mathbf{Q}$ PSD

Example 3, Data Fitting. (recall Handout#3)
Find the best least-squares data-fit, where we know a priori some linear relations among the parameters.

Example 4. In data mining, we construct a support vector machine by finding a hyperplane that gives the best "separation" between 2 data sets (the positive and negative examples).

Example 5, Markowitz's Investment Model. (H. Markowitz won the 1990 Nobel Prize in Economics for a model whose basics are what follows.)

We have historical performance data on n activities we can invest in. We want to invest in a mixture of these activities that intuitively has "maximum return & minimum risk". Markowitz models this by maximizing the objective function

(expectation of the return) $-\mu \times$ (variance of the return)

where μ is some multiplier.
Define

$$x_i = \text{the fraction of our investment that we'll put into activity } i$$
$$r_i = \text{the (historical) average return on investment } i$$
$$v_i = \text{the (historical) variance of investment } i$$
$$c_{ij} = \text{the (historical) covariance of investments } i \text{ \& } j$$

If I_i is the random variable equal to the return of investment i, our investment returns $\sum_i x_i I_i$. By elementary probability the variance of this sum is

$$\textstyle\sum_i x_i^2 v_i + 2\sum_{i<j} c_{ij} x_i x_j.$$

So forming $\mathbf{r}$, the vector of expected returns, & the covariance matrix $\mathbf{C} = (c_{ij})$, Markowitz's QP is

minimize $\mu\mathbf{x}^T\mathbf{C}\mathbf{x} - \mathbf{r}\mathbf{x}$
subject to $\mathbf{1}^T\mathbf{x} = 1$
$\mathbf{x} \geq \mathbf{0}$

Note that $v_i = c_{ii}$. Also the covariance matrix $\mathbf{C}$ is PSD, since the variance of a random variable is nonnegative.

Markowitz's model develops the family of solutions of the QP as μ varies
in some sense, these are the only investment strategies one should consider

the `LINDO` manual gives a similar example:
achieve a given minimal return ($\mathbf{r}\mathbf{x} \geq r_0$) while minimizing the variance

we can reduce many QPs to LP
intuition: in the small, the QP objective is linear

1. An optimum solution $\mathbf{x}^*$ to $\mathcal{Q}$ is also optimum to the LP

$$\begin{aligned} &\text{minimize } (\mathbf{c} + \mathbf{x}^{*T}\mathbf{Q})\mathbf{x} \\ &\text{subject to } \mathbf{A}\mathbf{x} \geq \mathbf{b} \\ &\qquad\qquad \mathbf{x} \geq \mathbf{0} \end{aligned}$$

Proof. take any feasible point and write it as $\mathbf{x}^* + \mathbf{\Delta}$
since $\mathbf{x}^*$ is optimum to $\mathcal{Q}$, its objective in $\mathcal{Q}$ is at most $\mathbf{c}(\mathbf{x}^* + \mathbf{\Delta}) + (\mathbf{x}^* + \mathbf{\Delta})^T\mathbf{Q}(\mathbf{x}^* + \mathbf{\Delta})/2$
thus

$$(\mathbf{c} + \mathbf{x}^{*T}\mathbf{Q})\Delta + \mathbf{\Delta}^T\mathbf{Q}\mathbf{\Delta}/2 \geq 0$$

since the feasible region is convex, $\mathbf{x}^* + \epsilon\mathbf{\Delta}$ is feasible, for any $0 \leq \epsilon \leq 1$
so the previous inequality holds if we replace Δ by $\epsilon\Delta$
we get an inequality of the form $a\epsilon + b\epsilon^2 \geq 0$, equivalently $a + b\epsilon \geq 0$
this implies $a \geq 0$

so $(\mathbf{c} + \mathbf{x}^{*T}\mathbf{Q})\Delta \geq 0$ as desired $\square$

Remark. the proof shows if $\mathbf{Q}$ is PD, $\mathcal{Q}$ has ≤ 1 optimum point
since $a \geq 0,\ b > 0 \Longrightarrow a + b > 0$

Example 1.
consider the QP for distance from the origin,

$$\begin{aligned} &\min\ q = x^2 + y^2 \\ &\text{s.t. } x + y \geq 1 \\ &\qquad x, y \geq 0 \end{aligned}$$

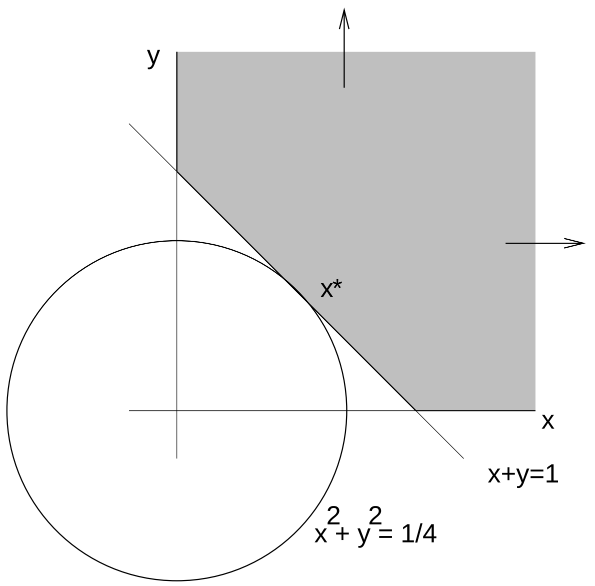


Fig.1 $\mathbf{x}^* = (1/2, 1/2)$ is the unique QP optimum.
$\mathbf{x}^*$ is not a corner point of the feasible region.

the cost vector of #1 is $\mathbf{c}' = \mathbf{x}^{*T}\mathbf{Q} = (1/2, 1/2)\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} = (1/2, 1/2)$
the linear cost function $\mathbf{c}'\mathbf{x} = x/2 + y/2$ approximates q close to $\mathbf{x}^*$

switching to cost $\mathbf{c}'\mathbf{x}$, $\mathbf{x}^*$ is still optimum
although other optima exist: the edge $E = \{(x, 1-x, 0) : 0 \le x \le 1\}$

Example 2:
modify the QP to

$$\begin{aligned} &\min\ q = z^2 + x + y\\ &\text{s.t. } x + y \ge 1\\ &\quad\ x, y, z \ge 0\end{aligned}$$

the set of optima is edge E

this is consistent with the Remark, since $\mathbf{Q} = \begin{bmatrix}0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 1\end{bmatrix}$ is PSD but not PD

2. Complementary slackness gives us conditions equivalent to optimality of the above LP:

Lemma. $\mathbf{x}$ *an optimum solution to* $\mathcal{Q} \Longrightarrow$ *there are column vectors* $\mathbf{u}$, $\mathbf{v}$, $\mathbf{y}$ *(of length* n, m, m *respectively) satisfying this LCP*:

$$\begin{bmatrix}\mathbf{u}\\ \mathbf{v}\end{bmatrix} - \begin{bmatrix}\mathbf{Q} & -\mathbf{A}^T\\ \mathbf{A} & \mathbf{0}\end{bmatrix}\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix} = \begin{bmatrix}\mathbf{c}^T\\ -\mathbf{b}\end{bmatrix}$$
$$\begin{bmatrix}\mathbf{u}\\ \mathbf{v}\end{bmatrix}^T\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix} = 0$$
$$\mathbf{x}, \mathbf{u}, \mathbf{y}, \mathbf{v} \ge \mathbf{0}$$

Proof.
the dual LP is
max $\mathbf{y}^T\mathbf{b}$ s.t. $\mathbf{y}^T\mathbf{A} \le \mathbf{c} + \mathbf{x}^{*T}\mathbf{Q}$, $\mathbf{y} \ge \mathbf{0}$

introducing primal (dual) slacks $\mathbf{v}$, $(\mathbf{u}^T)$,
the complementary slackness conditions for optimality are
$\mathbf{u}, \mathbf{v} \ge \mathbf{0}$, $\mathbf{u}^T\mathbf{x} = \mathbf{v}^T\mathbf{y} = 0$

the LCP expresses these conditions $\square$

the above LCP is the *Karush-Kuhn-Tucker necessary conditions (KKT conditions)* for optimality

taking $\mathbf{Q} = \mathbf{0}$ makes $\mathcal{Q}$ an LP
& the KKT conditions become the complementary slackness characterization of LP optimality

in fact we can think of the KKT conditions as nonnegativity
+ feasibility of the dual QP (condition on $\mathbf{c}^T$, see Handout#74)
+ primal feasibility (condition on $\mathbf{b}$)
+ complementary slackness

Theorem. *Let* $\mathbf{Q}$ *be PSD. Then*

$\mathbf{x}$ *is an optimum solution to* $\mathcal{Q} \iff \mathbf{x}$ *satisfies the KKT conditions.*

Proof.
$\Longleftarrow$: suppose $\mathbf{x}^*$ satisifies the KKT conditions
take any feasible point $\mathbf{x}^* + \mathbf{\Delta}$
the same algebra as above shows its objective in $\mathcal{Q}$ exceeds that of $\mathbf{x}^*$ by

$$(\mathbf{c} + \mathbf{x}^{*T}\mathbf{Q})\Delta + \mathbf{\Delta}^T\mathbf{Q}\mathbf{\Delta}/2$$

the first term is ≥ 0, since $\mathbf{x}^*$ is optimum to the LP of #1
this follows from the KKT conditions, which are complementary slackness for the LP
the second term is nonnegative, by PSD $\square$

Example: "KKT does Calc I"

we'll apply KKT in 1 dimension to optimize a quadratic function over an interval

problem: min $Ax^2 + Bx$ s.t. $0 \leq \ell \leq x \leq h$

Remarks.
1. we're minimizing a general quadratic $Ax^2 + Bx + C$ – the C disappears since it's irrelevant
2. for convenience we assume the interval's left end ℓ is nonnegative
3. really only the sign of A is important

our QP has $\mathbf{Q} = (2A),\ \mathbf{c} = (B),\ \mathbf{b} = (\ell, -h)^T,\ \mathbf{A} = (1, -1)^T$
so KKT is

$$\begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix} - \begin{bmatrix} 2A & -1 & 1 \\ 1 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} B \\ -\ell \\ h \end{bmatrix}$$
$$ux, v_1y_1, v_2y_2 = 0$$
$$\mathbf{x}, \mathbf{u}, \mathbf{y}, \mathbf{v} \geq \mathbf{0}$$

the linear constraints in scalar form:

$$\begin{aligned} u - 2Ax + y_1 - y_2 &= B \\ v_1 \quad - x \qquad\qquad &= -\ell \\ v_2 \quad + x \qquad\qquad &= h \end{aligned}$$

CS allows 3 possibilities, $v_1 = 0,\ v_2 = 0$, or $y_1 = y_2 = 0$

$v_1 = 0$: $\Longrightarrow x = \ell$
$v_2 = 0$: $\Longrightarrow x = h$
$y_1 = y_2 = 0$:
when $u = 0$: $\Longrightarrow -2Ax = B$ (i.e., first derivative $= 0$), $x = -B/2A$
when $u > 0$: $\Longrightarrow x = 0$, so $\ell = 0$, and $x = \ell$ as in first case

so KKT asks for the same computations as Calc's set-the-derivative-to-0 method

Lagrangian Multiplier Interpretation

Lagrangian relaxation tries to eliminate constraints
by bringing them into the objective function with a multiplicative penalty for violation

the Lagrangian for $\mathcal{Q}$ is
$\mathbf{L}(\mathbf{x},\mathbf{y},\mathbf{u}) = \mathbf{c}\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x} - \mathbf{y}^T(\mathbf{A}\mathbf{x} - \mathbf{b}) - \mathbf{u}^T\mathbf{x}$

the Lagrangian optimality conditions are:
feasibility: $\mathbf{A}\mathbf{x} \ge \mathbf{b}$, $\mathbf{x} \ge \mathbf{0}$
nonnegativity of multipliers: $\mathbf{y}, \mathbf{u} \ge \mathbf{0}$
no gain from feasibility: $(\mathbf{A}\mathbf{x})_i > b_i \Longrightarrow y_i = 0$; $x_j > 0 \Longrightarrow u_j = 0$
1st order optimality condition: $\frac{\partial L}{\partial \mathbf{x}} = \mathbf{0}$, i.e., $\mathbf{c}^T + \mathbf{Q}\mathbf{x} - \mathbf{A}^T\mathbf{y} - \mathbf{u} = \mathbf{0}$

Remarks.
1. the constraints preceding 1st order optimality ensure that
$\mathbf{L}(\mathbf{x},\mathbf{y},\mathbf{u})$ upper-bounds the objective function of $\mathcal{Q}$

2. the Lagrangian optimality conditions are exactly the KKT conditions

3. `LINDO` specifies a QP as an LP, in this form:
`min` $x_1 + \ldots + x_n + y_1 + \ldots + y_m$
`subject to`
1st order optimality constraints
$\mathbf{A}\mathbf{x} \ge \mathbf{b}$
`end`
`QCP` $n+2$

in the objective function, only the order of the variables is relevant
it specifies the order of the 1st order optimality conditions, & the order of the dual variables
the $\mathbf{u}$'s are omitted: the 1st order optimality conditions are written as inequalities
the `QCP` statement gives the row number of the first primal constraint $\mathbf{A}\mathbf{x} \ge \mathbf{b}$

many approximation algorithms for NP-hard problems are designed as follows:
formulate the problem as an ILP
relax the ILP to an LP by discarding the integrality constraints
solve the LP
use a "rounding procedure" to perturb the LP solution to a good integral solution

in the last 10 years a more powerful approach has been developed,
using semidefinite programming (SDP) instead of LP

here we model the problem by a general quadratic program
(achieve integrality using quadratic constraints)
relax by changing the variables to vectors
solve the relaxation as an SDP, and round

we illustrate by sketching Goemans & Williamson's approximation algorithm for MAX CUT

in the MAX CUT problem we're given an undirected graph G
we want a set of vertices S with the greatest number of edges joining S and $V - S$
more generally, each edge ij has a given nonnegative weight w_{ij}
& we want to maximize the total weight of all edges joining S and $V - S$
this problem can arise in circuit layout

General Quadratic Program
each vertex i has a variable $u_i \in \{1, -1\}$
the 2 possibilities for u_i correspond to the 2 sides of the cut

$$\text{so } u_i u_j = \begin{cases} 1 & i \text{ and } j \text{ are on the same side of the cut} \\ -1 & i \text{ and } j \text{ are on opposite sides of the cut} \end{cases}$$

it's easy to see MAX CUT amounts to this quadratic program:

$$\begin{array}{ll} \text{maximize } (1/2)\sum_{i<j} w_{ij}(1 - u_i u_j) & \\ \text{subject to } \; u_i^2 \; = \; 1 & \text{for each vertex } i \end{array}$$

next we replace the n variables u_i by n n-dimensional vectors $\mathbf{u}_i$
quadratic terms $u_i u_j$ become inner products $\mathbf{u}_i \cdot \mathbf{u}_j$

we get this "vector program":

$$\begin{array}{lrcll} \text{maximize } (1/2)\sum_{i<j} w_{ij}(1 - \mathbf{u}_i \cdot \mathbf{u}_j) & & & & \\ \text{subject to} & \mathbf{u}_i \cdot \mathbf{u}_i & = & 1 & \text{for each vertex } i \\ & \mathbf{u}_i & \in & \mathbf{R}^n & \text{for each vertex } i \end{array}$$

a cut gives a feasible solution using vectors $(\pm 1, 0, \ldots, 0)$
so this program is a relaxation of MAX CUT

for any n n-dimensional vectors $\mathbf{u}_i$, $i = 1, \ldots n$
form the $n \times n$ matrix $\mathbf{B}$ whose columns are the $\mathbf{u}_i$
then $\mathbf{X} = \mathbf{B}^T\mathbf{B}$ is PSD (Handout#64) with $x_{ij} = \mathbf{u}_i \cdot \mathbf{u}_j$
furthermore, any symmetric PSD $\mathbf{X}$ arises in this way

thus our vector program is equivalent to the following program:

SDP

$$\begin{array}{lrcll} \text{maximize } (1/2)\sum_{i<j} w_{ij}(1-x_{ij}) & & & & \\ \text{subject to} & x_{ii} & = & 1 & \text{for each vertex } i \\ & x_{ij} & = & x_{ji} & \text{for each } i \neq j \\ & (x_{ij}) & \text{is} & PSD & \end{array}$$

this is a semidefinite program (Handout#41)

so we can solve it (to within arbitrarily small additive error) in polynomial time

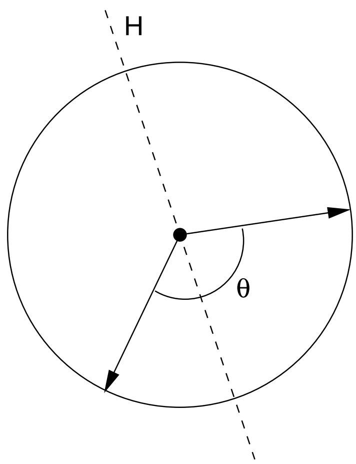
the vectors $\mathbf{u}_i$ can be computed from (x_{ij}) (to within any desired accuracy)
in polynomial time (Handout#64)

so we can assume we have the vectors $\mathbf{u}_i$; now we need to round them to values ± 1

in the rest of the discussion let $\mathbf{u}_i$ & $\mathbf{u}_j$ be 2 arbitrary vectors
abbreviate them to $\mathbf{u}$ & $\mathbf{u}'$, and abbreviate w_{ij} to w
also let θ be the angle between $\mathbf{u}$ & $\mathbf{u}'$

recall the definition of scalar product (Handout#65)
then our 2 vector contribute $(w/2)(1-\cos\theta)$ to the objective function
the bigger the angle θ, the bigger the contribution
so we should round vectors with big angles to opposite sides of the cut

Rounding Algorithm. Let H be a random hyperplane through the origin in $\mathbf{R}^n$. Round all vectors on the same side of H to the same side of the cut. (Vectors on H are rounded arbitrarily.)



Random hyperplane H separating 2 unit vectors at angle θ.

generating H in polynomial time is easy, we omit the details the only remaining question is, how good an approximation do we get?

let OPT denote the maximum weight of a cut
let z^* be the optimum value of the SDP (so $z^* \geq$ OPT)
let EC be the expected weight of the algorithm's cut
the (expected worst-case) *approximation ratio* is the smallest possible value of $\alpha =$ EC/OPT
so $\alpha \geq$ EC$/z^*$

linearity of expectations shows EC is the sum, over all pairs i, j,
of the expected contribution of edge ij to the cut's weight
so we analyze the expected contribution of our 2 typical vectors $\mathbf{u}$, $\mathbf{u}'$

the probability that $\mathbf{u}$ & $\mathbf{u}'$ round to opposite sides of the cut is exactly θ/π (see the figure)
$\therefore$ the contribution of this pair to EC is $w\theta/\pi$

then $\alpha \geq \min_{0\leq\theta\leq\pi} \dfrac{w\theta/\pi}{(w/2)(1-\cos\theta)} = \dfrac{2}{\pi} \min_{0\leq\theta\leq\pi} \dfrac{\theta}{1-\cos\theta}$

calculus shows the last expression is $> .878$
to simplify the calculation, verify the identity $2\theta/\pi > .878(1-\cos\theta)$

Conclusion. *The SDP algorithm has approximation ratio* $> .878$.

Minimum Cost Network Flow

a network has "sites" interconnected by "links"
material circulates through the network
transporting material across the link from site i to site j costs c_{ij} dollars per unit of material
the link from i to j must carry $\geq \ell_{ij}$ units of material and $\leq u_{ij}$ units
find a minimum cost routing of the material

letting x_{ij} be the amount of material shipped on link ij gives this LP:

$$\begin{array}{lrll} \text{minimize } z = & \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} & & \\ \text{subject to} & \sum_{i=1}^{n} x_{ij} - \sum_{k=1}^{n} x_{jk} = 0 & j = 1, \ldots, n & \textit{flow conservation} \\ & x_{ij} \geq \ell_{ij} & i, j = 1, \ldots, n & \\ & x_{ij} \leq u_{ij} & i, j = 1, \ldots, n & \end{array}$$

Some Variations

Networks with Losses & Gains

a unit of material starting at i gets multiplied by m_{ij} while traversing link ij
so replace conservation by $\sum_{i=1}^{n} m_{ij} x_{ij} - \sum_{k=1}^{n} x_{jk} = 0$

example from currency conversion:
a "site" is a currency, e.g., dollars, pounds
m_{ij} = the number of units of currency j purchased by 1 unit of currency i
sample problem: convert \$10000 into the most rubles possible

more examples: investments at points in time (\$1 now $\rightarrow$ \$1.08 in a year), conversion of raw materials into energy (coal $\rightarrow$ electricity), transporting materials (evaporation, seepage)

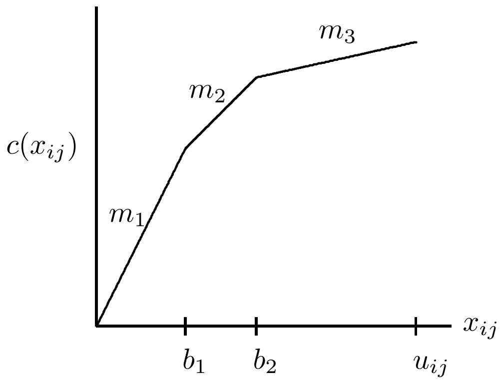
Multicommodity Flow
1 network transports flows of various types (shipping corn, wheat, etc.)
use variables x_{ij}^{k}, for k ranging over the commodities

if we're routing people, Internet packets or telephone messages, we get an ILP (x_{ij}^{k} integral)

in the next 2 examples take $\ell_{ij} = 0$ (for convenience)

Concave Down Cost Functions (works for any LP)

for convenience assume we're maximizing z = profit, not minimizing cost
the profit of transporting material on link ij is a piecewise linear concave down function:



"decreasing returns to scale"

replace x_{ij} by 3 variables r_{ij}, s_{ij}, t_{ij}
each appears in the flow conservation constraints where x_{ij} does
bounds on variables: $0 \le r_{ij} \le b_1$, $0 \le s_{ij} \le b_2 - b_1$, $0 \le t_{ij} \le u_{ij} - b_2$
objective function contains terms $m_1 r_{ij} + m_2 s_{ij} + m_3 t_{ij}$

concavity of $c(x_{ij})$ ensures this is a correct model

Fixed Charge Flow

link ij incurs a "startup cost" s_{ij} if material flows across it
ILP model:
introduce *decision variable* $y_{ij} = 0$ or 1
new upper bound constraint: $x_{ij} \le u_{ij} y_{ij}$
objective function: add term $s_{ij} y_{ij}$

Assignment Problem

there are n workers & n jobs
assigning worker i to job j costs c_{ij} dollars
find an assignment of workers to jobs with minimum total cost

let x_{ij} be an indicator variable for the condition, worker i is assigned to job j
we get this LP:

$$\text{minimize } z = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$$

$$\begin{array}{llll} \text{subject to} & \sum_{j=1}^{n} x_{ij} & = 1 & i \ = \ 1, \ldots, n \\ & \sum_{i=1}^{n} x_{ij} & = 1 & j \ = \ 1, \ldots, n \\ & x_{ij} & \ge 0 & i, j = 1, \ldots, n \end{array}$$

x_{ij} should be constrained to be integral
but the optimum always occurs for an integal x_{ij}
so we solve the ILP as an LP!

Set Covering

constructing fire station j costs c_j dollars, $j = 1, \ldots, n$
station j could service some known subset of the buildings
construct a subset of the n stations so each building can be serviced
and the cost is minimum

let $a_{ij} = 1$ if station j can service building i, else 0
let x_j be an indicator variable for construcing station j

$$\begin{array}{lll} \text{minimize } z = & \sum_{j=1}^n c_j x_j & \\ \text{subject to} & \sum_{j=1}^n a_{ij} x_j \ge 1 & i = 1, \ldots, m \\ & x_j \ge 0, \text{ integral} & j = 1, \ldots, n \end{array}$$

this ILP is a *set covering problem* –
choose sets from a given family, so each element is "covered", minimizing total cost

similarly we have
set packing problem – choose disjoint sets, maximizing total cost
set partitioning problem – choose sets so every element is in exactly one set

Facility Location

elaborates on the above fire station location problem –
there are m clients and n potential locations for facilities
we want to open a set of facilities to service all clients, minimizing total cost
e.g., post offices for mail delivery, web proxy servers

constructing facility j costs c_j dollars, $j = 1, \ldots, n$
facility j services client i at cost of s_{ij} dollars

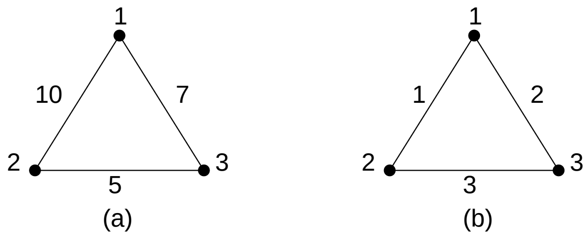
ILP model:
let x_j be an indicator variable for opening facility j
let y_{ij} be an indicator variable for facility j servicing client i

$$\begin{array}{lll} \text{minimize } z = & \sum_j c_j x_j + \sum_{i,j} s_{ij} y_{ij} & \\ \text{subject to} & \sum_j y_{ij} = 1 & i \text{ a client} \\ & y_{ij} \le x_j & i \text{ a client, } j \text{ a facility} \\ & y_{ij}, x_j \in \{0,1\} & i \text{ a client, } j \text{ a facility} \end{array}$$

this illustrates how ILP models "if then" constraints

Quadratic Assignment Problem

there are n plants and n locations
we want to assign each plant to a distinct location
each plant p ships s_{pq} units to every other plant q
the per unit shipping cost from location i to location j is c_{ij}
find an assignment of plants to locations with minimum total cost



Quadratic assignment problem.
(a) Amounts shipped between 3 plants.
(b) Shipping costs for 3 locations.
Optimum assignment = identity, cost $10 \times 1 + 7 \times 2 + 5 \times 3 = 39$.

let x_{ip} be an indicator variable for assigning plant p to location i
set $d_{ijpq} = c_{ij} s_{pq}$

$$
\begin{array}{lll}
\text{minimize } z = & \sum_{i,j,p,q} d_{ijpq} x_{ip} x_{jq} & \\
\text{subject to} & \sum_{p=1}^{n} x_{ip} = 1 & i = 1,\ldots,n \\
& \sum_{i=1}^{n} x_{ip} = 1 & p = 1,\ldots,n \\
& x_{ip} \in \{0,1\} & i,p = 1,\ldots,n
\end{array}
$$

Remarks.

1. we could convert this to an ILP–
   introduce variables $y_{ijpq} \in \{0,1\}$, & force them to equal $x_{ip}x_{jq}$ by the constraint
   $$y_{ijpq} \ge x_{ip} + x_{jq} - 1$$

   but this adds many new variables & constraints

2. a *quadratic program* has the form

$$\begin{array}{lll}
\text{maximize } z = & \sum_{j,k} c_{jk} x_j x_k + \sum_j c'_j x_j & \\
\text{subject to} & \sum_{j=1}^{n} a_{ij} x_j \le b_i & i = 1, \ldots, m
\end{array}$$

3. we can find a feasible solution to an ILP

$$\begin{array}{lll}
\text{maximize } z = & \sum_j c_j x_j & \\
\text{subject to} & \sum_{j=1}^{n} a_{ij} x_j \le b_i & i = 1, \ldots, m \\
 & x_j \in \{0,1\} & j = 1, \ldots, n
\end{array}$$

   by solving the QP

$$\begin{array}{lll}
\text{maximize } z = & \sum_j (x_j - 1/2)^2 & \\
\text{subject to} & \sum_{j=1}^{n} a_{ij} x_j \le b_i & i = 1, \ldots, m \\
 & x_j \le 1 & j = 1, \ldots, n \\
 & x_j \ge 0 & j = 1, \ldots, n
\end{array}$$

Example
we want to maximize profit $500p + 620f$ from producing pentium (p) & 486 (f) computer systems
but maintaining a high-tech image dictates maximize p

whatever the strategy it should be a *vector maximum*, ("pareto optimum") i.e.,
if we produce (p, f) units, no other feasible schedule (p', f') should have $p' \geq p$ & $f' \geq f$

we give several approaches to multiobjective problems

a common aspect is that in practice, we iterate the process
resolving the LP with different parameters until a satisfactory solution is obtained
sensitivity analysis techniques (Handout #35) allow us to efficiently resolve an LP
if we modify it in a way that the solution changes "by a small amount"

now write our objective functions as $f_k = \sum_j c_{kj} x_j + d_k,\ k = 1, \ldots, r$

Prioritization (lexicographic optimum)
index the objective functions in order of decreasing importance $f_1, \ldots, f_k$

solve the LP using objective maximize $z = f_1$
let z_1 be the maximum found
if the optimum solution vector $(x_1, \ldots, x_n)$ is unique, stop
otherwise
add the constraint $\sum_j c_{1j} x_j + d_1 = z_1$
repeat the process for f_2
keep on repeating for $f_3, \ldots, f_r$
until the optimum is unique or all objectives have been handled

Remarks
1. the optimum is unique if every nonbasic cost coefficient is negative
the possibility that a nonbasic cost coefficient is 0 prevents this from being iff

2. sensitivity analysis allow us to add a new constraint easily

Worst-case Approach

optimize the minimax value of the objectives
(as in Handout #3)

Weighted Average Objective

solve the LP with objective $\sum_k w_k f_k$
where the weights w_k are nonnegative values summing to 1

if the solution is unreasonable, adjust the weights and resolve
starting the simplex from the previous optimum will probably be very efficient

Goal Programming

adapt a goal value g_k for each objective function f_k
and use appropriate penalities for excess & shortages of each goal

e.g., $p_e = p_s = 1$ keeps us close to the goal
$p_e = 0$, $s_e = 1$ says exceeding the goal is OK but each unit of shortfall incurs a unit penalty

iterate this process, varying the parameters, until a satisfactory solution is achieved

Goal Setting with Marginal Values

choose a primary objective function f_0 and the other objectives f_k, $k = 1, \ldots, r$
f_0 is most naturally the monetary price of the solution

adapt goals g_k, $k = 1, \ldots, r$
solve the LP with objective $z = f_0$ and added constraints $f_k = g_k, k = 1, \ldots, r$
duality theory (Handout #20) reveals the *price* p_k of each goal g_k:
changing g_k by a small ϵ changes the cost by ϵp_k
use these prices to compute better goals that are achieved at an acceptable cost
resolve the LP to verify the predicted change
iterate until the solution is satisfactory

this handout proves the assertions of Handout#10,p.3

consider a standard form LP $\mathcal{L}$, with P the corresponding convex polyhedron
  $P \subseteq \mathbf{R}^n$, activity space, i.e., no slacks

we associate each (decision or slack) variable of $\mathcal{L}$ with a unique constraint of $\mathcal{L}$:
  the constraint for x_j is
$$\begin{cases} \text{nonnegativity} & \text{if } x_j \text{ is a decision variable} \\ \text{the corresponding linear inequality} & \text{if } x_j \text{ is a slack variable} \end{cases}$$

(minor point: no variable is associated with the nonnegativity constraint of a slack variable)

a variable $= 0 \iff$ its constraint holds with equality

Fact 1. *A bfs is a vertex* $\mathbf{x}$ *of* P *plus* n *hyperplanes of* P *that uniquely define* $\mathbf{x}$.
$(*)$ *The constraints of the nonbasic variables are the* n *hyperplanes that define* $\mathbf{x}$.

Proof.
consider a bfs $\mathbf{x}$, and its corresponding dictionary D (there may be more than 1)
when the nonbasic variables are set to 0, $\mathbf{x}$ is the unique solution of D
hence $\mathbf{x}$ is the unique point on the hyperplanes of the nonbasic variables
  (since D is equivalent to the initial dictionary, which in turn models $\mathcal{L}$)
so we've shown a bfs gives a vertex, satisfying $(*)$

conversely, we'll show that any vertex of P corresponds to a dictionary, satisfying $(*)$
take n hyperplanes of P that have $\mathbf{x}$ as their unique intersection
let N be the variables that correspond to these n hyperplanes
let B be the remaining m variables

set the variables of N to 0
this gives a system of n equations with a unique solution, $\mathbf{x}$

let's reexpress this fact using matrices:
write LP $\mathcal{L}$ in the equality form $\mathbf{Ax} = \mathbf{b}$ of Handout#23,p.1
then $\mathbf{A}_B\mathbf{x} = \mathbf{b}$ has a unique solution

this shows the matrix $\mathbf{A}_B$ is nonsingular (Handout#55,p.2)
thus the Theorem of Handout#23,p.2 shows B is a basis
  the nonbasic variables N are described by $(*)$ $\square$

Fact 2. *A nondegenerate pivot moves from one vertex, along an edge of* P*, to an adjacent vertex.*

Proof.
a pivot step travels along a line segment whose equation, in parameterized form,
  is given in Handout #8, Property 5:

$$x_j = \begin{cases} t & j = s \\ b_j - a_{js}t & j \in B \\ 0 & j \notin B \cup s \end{cases}$$

let t range from $-\infty$ to ∞ in this equation to get a line ℓ

in traversing ℓ, the $n-1$ variables other than $B \cup s$ remain at 0
thus ℓ lies in each of the $n-1$ corresponding hyperplanes

in fact the dictionary shows ℓ is exactly equal to the intersection of these $n-1$ hyperplanes

so the portion of ℓ traversed in the pivot step is an edge of P $\square$

the last fact is a prerequisite for Hirsch's Conjecture:

Fact 3. *Any 2 vertices of a convex polyhedron are joined by a simplex path.*

Proof.
let $\mathbf{v}$ & $\mathbf{w}$ be any 2 vertices of P

Claim. there is a cost function $\sum_{j=1}^n c_j x_j$ that is tangent to P at $\mathbf{w}$
i.e., the hyperplane $\sum_{j=1}^n c_j x_j = \sum_{j=1}^n c_j w_j$ passes thru $\mathbf{w}$, but thru no other point of P

the Claim implies Fact 3:
execute the simplex algorithm on the LP $\mathcal{L}$ with the Claim's objective function
choose the initial dictionary to correspond to vertex $\mathbf{v}$
simplex executes a sequence of pivots that end at $\mathbf{w}$ (since $\mathbf{w}$ is the only optimum point)
this gives the desired simplex path

Proof of Claim.
write every constraint of $\mathcal{L}$ in the form $\sum_{j=1}^n a_{ij} x_j \le b_i$
so a nonnegativity constraint $x_j \ge 0$ becomes $-x_j \le 0$

let $\mathbf{w}$ be the unique intersection of n hyperplanes of P,
corresponding to constraints $\sum_{j=1}^n a_{ij} x_j \le b_i$, $i = 1, \ldots, n$
(some of these may be nonnegativity)

take $c_j = \sum_{i=1}^n a_{ij}$
every point $\mathbf{x} \in P - \mathbf{w}$ has $\sum_{j=1}^n c_j x_j < \sum_{j=1}^n c_j w_j$
since $\mathbf{w}$ satisfies the n constraints with $=$
& every other point of P has $<$ in at least 1 of these constraints $\square$

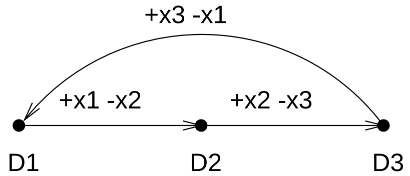
(see Handout#53 for a deeper look at the Claim)

Corollary. *Any face of P is the set of optimum solutions for some objective function.*
Proof. as above, use the hyperplanes of the face to construct the objective function $\square$

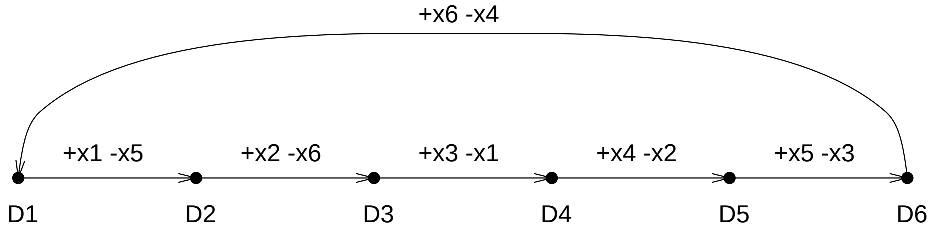
the converse of this statement is proved in the first exercise of Handout#19

Simplex Cycles

the simplest example of cycling in the simplex algorithm
has the variables swapped in and out in a fixed cyclic order



Chvátal's example of cycling (pp.31–32) is almost as simple:



but cycles in the simplex algorithm can be exponentially long!

e.g., a cycle can mimic a Gray code

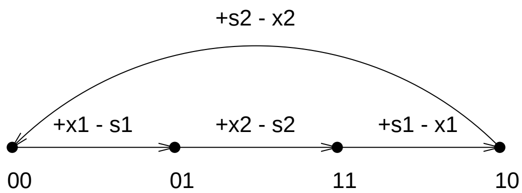
a *Gray code* is a sequential listing of the 2^n possible bitstrings of n bits, such that
each bitstring (including the 1st) differs from the previous by flipping 1 bit

Examples:
(i) $00, 01, 11, 10$

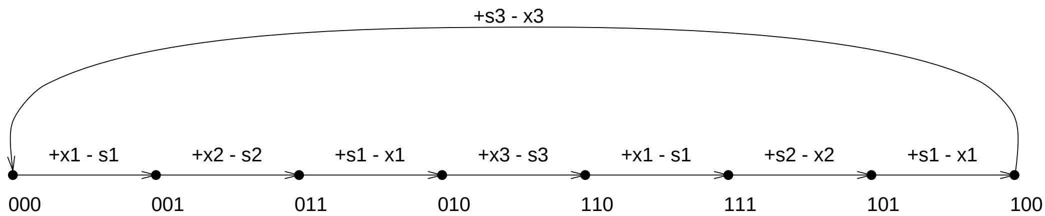
(ii) G_n is a specific Gray code on n bits, from $0 \ldots 0$ to $10 \ldots 0$:
recursive recipe for G_n:
start with $0G_{n-1}$ $(00 \ldots 0 \to \ldots \to 01 \ldots 0)$
flip the leftmost bit to 1 $(\to 110 \ldots 0)$
do $1G_{n-1}$ in reverse $(110 \ldots 0 \to 100 \ldots 0)$

e.g., example (i) is G_2, and G_3 is
$000, 001, 011, 010, 110, 111, 101, 100$

G_k gives a simplex cycle of 2^k pivots involving only $2k$ variables, e.g.,



simplex cycle for $k = 2$



simplex cycle for $k = 3$

geometrically the Gray code G_n is a Hamiltonian tour
of the vertices of the n-dimensional unit cube

Klee-Minty Examples

on these LPs the simplex algorithm takes $2^n - 1$ pivots to find the optimum
the feasible region is a (perturbed) n-dimensional unit cube
so the standard form LP has n variables and n constraints

the pivot sequence is the above sequence derived from G_n
$00\ldots0 \to \ldots \to 01\ldots0 \to 110\ldots0 \to 100\ldots0$
initial bfs ... optimum bfs

you can check this using Problem 4.3 (Chvátal, p.53). it says
after $2^{n-1} - 1$ pivots x_{n-1} is the only basic decision variable
this corresponds to $010\ldots0$
after $2^n - 1$ pivots x_n is the only basic decision variable
this corresponds to $100\ldots0$

Klee-Minty examples have been devised for most known pivot rules

the smallest-subscript rule can do an exponential number of pivots before finding the optimum
in fact it can <u>stall</u> for an exponential number of pivots!

to understand stalling we'll redo the proof of Handout #12:

consider a sequence $\mathcal{S}$ of degenerate pivots using the smallest-subscript rule
so the bfs never changes in $\mathcal{S}$

say a pivot step *involves* the entering & leaving variables, but no others

a variable x_i is *fickle* if it's involved in > 1 pivot of $\mathcal{S}$

if there are no fickle variables, $|\mathcal{S}| \le n/2$

suppose $\mathcal{S}$ has fickle variables; let t be the largest subscript of a fickle variable

Corollary. *$\mathcal{S}$ has a nonfickle variable which is involved in a pivot between the first two pivots involving x_t.*

Proof. (essentially same argument Handout #12)

let F be the set of subscripts of fickle variables

let D & D^* be the dictionaries of the first two pivot steps involving x_t, with pivots as follows:

$$\underset{D}{\bullet}\xrightarrow{\;x_s - x_t\;} \quad \cdots \quad \underset{D^*}{\bullet}\xrightarrow{\;x_t - \cdot\;}$$

D may precede D^* in $\mathcal{S}$ or vice versa

as in Handout #12, $c_s = c_s^* - \sum_{i\in B} c_i^* a_{is}$ $\qquad (*)$

$c_s > 0$: since x_s is entering in D's pivot

we can assume $c_s^* \le 0$:
suppose $c_s^* > 0$
$\Longrightarrow x_s$ is nonbasic in D^*
$s > t$ (since x_s doesn't enter in D^*'s pivot)
$\Longrightarrow x_s$ isn't fickle
so D's pivot proves the Corollary!
(note: D^* precedes D in this case)

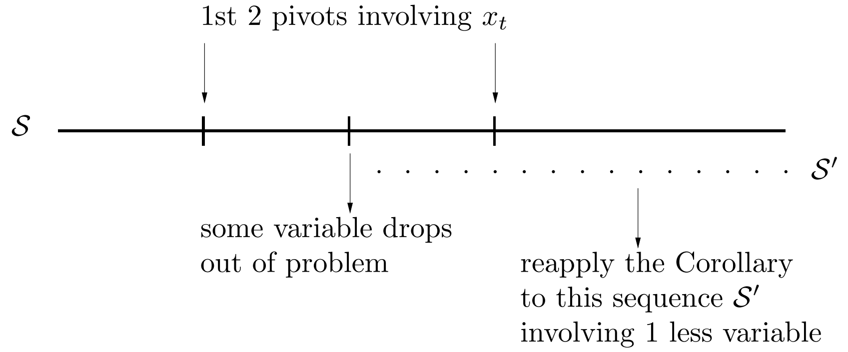
$c_i^* a_{is} \ge 0$ for $i \in B \cap F$:

Case $i = t$:
$a_{ts} > 0$: since x_t is leaving in D's pivot
$c_t^* > 0$: since x_t is entering in D^*'s pivot

Case $i \in B \cap (F - t)$:
$a_{is} \le 0$: $b_i = 0$ (since $x_i = 0$ throughout $\mathcal{S}$)
 but x_i isn't the leaving variable in D's pivot
$c_i^* \le 0$: otherwise x_i is nonbasic in D^* & D^*'s pivot makes x_i entering ($i < t$)

since the r.h.s. of $(*)$ is positive, some $i \in B - F$ has $c_i^* \neq 0$
hence x_i is in B but not B^*, i.e., a pivot between D & D^* involves x_i $\square$

we can apply the Corollary repeatedly to reveal the structure of $\mathcal{S}$:



starting with n variables, $\le n$ can drop out
so eventually there are no fickle variables
i.e., the smallest subscript rule never cycles

but $\mathcal{S}$ can have exponential length
 the recursive nature of this picture is a guide to constructing a bad example

Stalling Example

Chvátal (1978) gave the following Klee-Minty example:

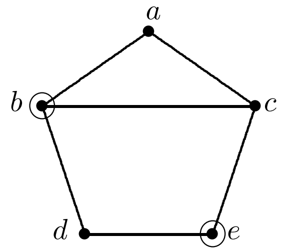
let ϵ be a real number $0 < \epsilon < 1/2$
consider the LP
 maximize $\sum_{j=1}^{n} \epsilon^{n-j} x_j$
 subject to $2(\sum_{j=1}^{i-1} \epsilon^{i-j} x_j) + x_i + x_{n+i} = 1, \qquad i = 1, \ldots, n$
 $x_j \ge 0, \qquad j = 1, \ldots, 2n$

start with the bfs $(0, \ldots, 0, 1, \ldots, 1)$ of n 0's & n 1's
& use Bland's rule

it takes f_n pivots to reach the optimum
where f_n is defined by
 $f_1 = 1,\ f_2 = 3,\ f_n = f_{n-1} + f_{n-2} - 1$
& $f_n \ge$ (the nth Fibonacci number) $\ge 1.6^{n-2}$

a minor variant of this LP does exactly the same pivots <u>at the origin</u>
 i.e., it stalls for an exponential number of pivots
& then does 1 pivot to the optimum

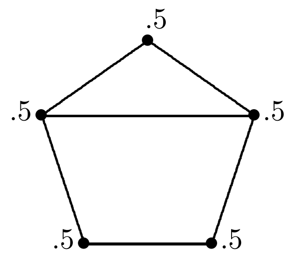
Example Graph & ILPs



graph & maximum independent set

maximize $x_a + x_b + x_c + x_d + x_e$
subject to $x_a + x_b \leq 1$
$x_a + x_c \leq 1$
$x_b + x_c \leq 1$
$x_b + x_d \leq 1$
$x_c + x_e \leq 1$
$x_d + x_e \leq 1$
$x_a, x_b, x_c, x_d, x_e \in \{0, 1\}$

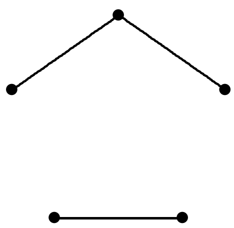
maximum independent set ILP



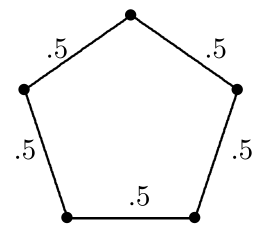
maximum fractional independent set

minimize $y_{ab} + y_{ac} + y_{bc} + y_{bd} + y_{ce} + y_{de}$
subject to $y_{ab} + y_{ac} \geq 1$
$y_{ab} + y_{bc} + y_{bd} \geq 1$
$y_{ac} + y_{bc} + y_{ce} \geq 1$
$y_{bd} + y_{de} \geq 1$
$y_{ce} + y_{de} \geq 1$
$y_{ab}, y_{ac}, y_{bc}, y_{bd}, y_{ce}, y_{de} \in \{0, 1\}$

minimum edge cover ILP



minimum edge cover



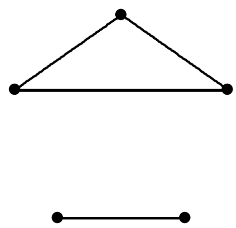
minimum fractional edge cover

maximize $x_a + x_b + x_c + x_d + x_e$
subject to $x_a + x_b + x_c \leq 1$
$x_b + x_d \leq 1$
$x_c + x_e \leq 1$
$x_d + x_e \leq 1$
$x_a, x_b, x_c, x_d, x_e \in \{0, 1\}$

maximum independent set ILP (clique constraints)

minimize $y_{abc} + y_{bd} + y_{ce} + y_{de}$
subject to $y_{abc} \geq 1$
$y_{abc} + y_{bd} \geq 1$
$y_{abc} + y_{ce} \geq 1$
$y_{bd} + y_{de} \geq 1$
$y_{ce} + y_{de} \geq 1$
$y_{abc}, y_{bd}, y_{ce}, y_{de} \in \{0, 1\}$

minimum clique cover ILP



minimum clique cover

Independent Sets & Duality

consider an undirected graph
an *independent set* is a set of vertices, no two of which are adjacent
a *maximum independent set* contains the greatest number of vertices possible

finding a maximum independent set is an NP-complete problem

we can formulate the problem as an ILP:
each vertex j has a 0-1 variable x_j
  $x_j = 1$ if vertex j is in the independent set, 0 if it is not

maximum	maximize $z = \sum_{j=1}^{n} x_j$	
independent	subject to $x_j + x_k \le 1$	(j,k) an edge of G
set ILP:	$x_j \in \{0,1\}$	$j = 1,\ldots,n$

LP	maximize $z = \sum_{j=1}^{n} x_j$	
relaxation:	subject to $x_j + x_k \le 1$	(j,k) an edge of G
	$x_j \ge 0$	$j = 1,\ldots,n$

(the LP relaxation needn't constrain $x_j \le 1$)

number the edges of G from 1 to m

LP	minimize $z = \sum_{i=1}^{m} y_i$	
dual:	subject to $\sum\{y_i : \text{vertex } j \text{ is on edge } i\} \ge 1$	$j = 1,\ldots,n$
	$y_i \ge 0$	$i = 1,\ldots,m$

integral	minimize $z = \sum_{i=1}^{m} y_i$	
dual:	subject to $\sum\{y_i : \text{vertex } j \text{ is on edge } i\} \ge 1$	$j = 1,\ldots,n$
	$y_i \in \{0,1\}$	$i = 1,\ldots,m$

(constraining y_i integral is the same as making it 0-1)

an *edge cover* of a graph is a set of edges spanning all the vertices
a *minimum edge cover* contains the fewest number of edges possible

the integral dual is the problem of finding a minimum edge cover
  $y_i = 1$ if edge i is in the cover, else 0

Weak Duality implies
  (size of a maximum independent set) $\le$ (size of a minimum edge cover)
indeed this is obvious – each vertex of an independent set requires its own edge to cover it

we can find a minimum edge cover in polynomial time
  thus getting a bound on the size of a maximum independent set

A Better Upper Bound

a *clique* of a graph is a complete subgraph, i.e., a set of vertices joined by every possible edge

an independent set contains ≤ 1 vertex in each clique
this gives an ILP with more constraints:

clique	maximize $z = \sum_{j=1}^{n} x_j$	
constraint	subject to $\sum\{x_j : \text{vertex } j \text{ is in } C\} \le 1$	C a maximal clique of G
MIS ILP:	$x_j \in \{0,1\}$	$j = 1,\ldots,n$

this can be a large problem –
the number of maximal cliques can be exponential in n !
the payoff is the LP solution will be closer to the ILP

LP relaxation: last line becomes $x_j \ge 0$

dual	minimize $z = \sum\{y_C : C \text{ a maximal clique of } G\}$	
LP:	subject to $\sum\{y_C : \text{vertex } j \text{ is in } C\} \ge 1$	$j = 1,\ldots,n$
	$y_C \ge 0$	C a maximal clique of G

integral dual LP: $y_C \in \{0,1\}$

a *clique cover* is a collection of cliques that spans every vertex
the integral dual LP is the problem of finding a minimum clique cover

Weak Duality:
(size of a maximum independent set) $\le$ (size of a minimum clique cover)

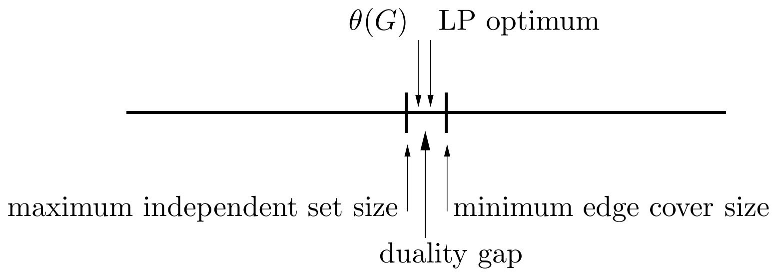
the rest of this handout assumes we use the clique constraint ILP for maximal independent sets

a graph is *perfect* if the relaxed LP is an integral polyhedron
equivalently, G, and all its induced subgraphs,
achieve equality in ILP Weak Duality (Chvátal, 1975)

there are many families of perfect graphs:
bipartite graphs, interval graphs, comparability graphs, triangulated (chordal) graphs, ...

a maximum independent set of a perfect graph can be found in polynomial time
(Grötschel, Lovasz, Schrijver, 1981)

the Lovász number $\theta(G)$ lies in the duality gap of the two ILPs
for a perfect graph $\theta(G)$ is the size of a maximum independent set
$\theta(G)$ is computable in polynomial time (GLS)



Generalization to Hypergraphs

consider a family $\mathcal{F}$ of subsets of V
a *packing* is a set of elements of V, ≤ 1 in each set of $\mathcal{F}$
a *covering* is a family of sets of $\mathcal{F}$ collectively containing all elements of V

maximum packing ILP:
maximize $\sum_{j=1}^{n} x_j$ subject to $\sum\{x_j : j \in S\} \le 1$, $S \in \mathcal{F}$; $x_j \in \{0,1\}$, $j = 1,\ldots,n$

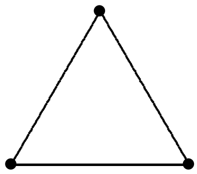
minimum covering ILP:
minimize $\sum_{S\in\mathcal{F}} y_S$ subject to $\sum\{y_S : j \in S\} \ge 1$, $j = 1,\ldots,n$; $y_S \in \{0,1\}$, $S \in \mathcal{F}$

as above, the LP relaxations of these 2 ILPs are duals, so any packing is $\le$ any covering

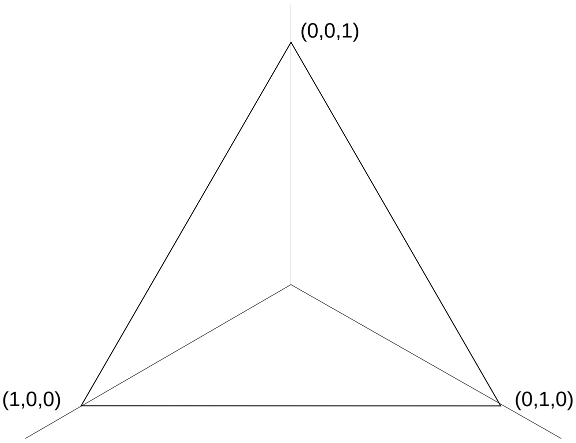
this packing/covering duality is the source of a number of beautiful combinatoric theorems
where the duality gap is 0
in these cases the ILPs are solvable in polynomial time!
e.g., finding a maximum flow; packing arborescences in a directed graph

Perfect Graph Example

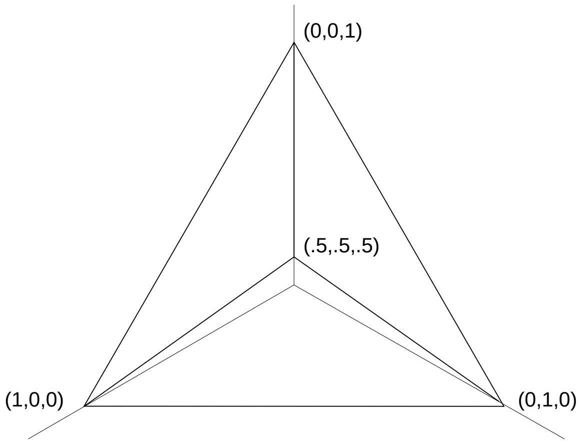
K_3 is the triangle:



here are the MIS polyhedra for K_3:



clique constraints



edge constraints

the clique constraints give an integral polyhedron
so K_3 is perfect

observe that the vertices of this polyhedron correspond to the independent sets of K_3
(more precisely, the vertices are the characteristic vectors of the independent sets)
this holds in general:

Theorem. *Take any graph, & its MIS polyhedron P*
defined by edge constraints or clique constraints.
P is an integral polyhedron $\Longleftrightarrow$
its vertices are precisely the independent sets of G.

Proof.
$\Longleftarrow$: trivial (the characteristic vector of an independent set is 0-1)

$\Longrightarrow$: the argument consists of 2 assertions:
(i) every independent set is a vertex of P
(ii) every vertex of P is an independent set

for simplicity we'll prove the assertions for the edge constraints
the same argument works for the clique constraints

Proof of (i)

let I be an independent set, with corresponding vector (x_i)
$x_i = 1$ if $i \in I$ else 0

choose n constraints (that (x_i) satisfies with equality) as follows:
for $i \notin I$ choose nonnegativity, $x_i \geq 0$
for $i \in I$ choose the constraint for an edge containing i
no 2 i's choose the same edge constraint, since I is independent
(x_i) satisfies these n constraints with equality, & no other point of P does:
each chosen edge constraint has 1 end constrained to 0
so the other end *must* equal 1

Proof of (ii)

a vertex (x_i) is a 0-1 vector, by nonnegativity and the edge constraints

if $x_i = x_j = 1$ then (i, j) is not an edge (since (x_i) is feasible)
thus (x_i) corresponds to an independent set $\square$

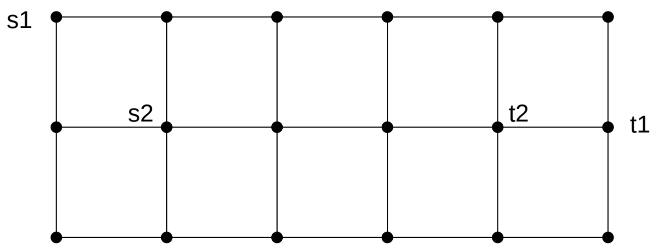
Polyhedral Combinatorics

to analyze the independent sets of a graph G,
we can analyze the polyhedron P whose vertices are those independent sets

this depends on having a nice description of P
which we have if G is perfect

in general polyhedral combinatorics analyzes a family of sets
by analyzing the polyhedron whose vertices are (the characteristic vectors of) those sets

Disjoint Paths Problem: Given a graph, vertices s, t & integer k, are there k openly disjoint st-paths?



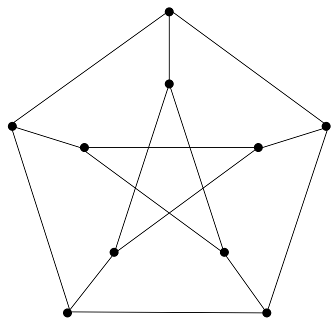
Example graph G_0 has 2 openly disjoint s_1t_1-paths
& 3 openly disjoint s_2t_2-paths

Disjoint Paths Problem is in $\mathcal{P}$ (i.e., it has a polynomial-time algorithm) because of this min-max theorem:

Menger's Theorem. *For any 2 nonadjacent vertices s, t, the greatest number of openly disjoint st-paths equals the fewest number of vertices that separate s and t.*

Hamiltonian Cycle Problem: Does a given graph have a tour passing through each vertex exactly once?

e.g., G_0 has a Hamiltonian cycle



The Peterson graph has no Hamiltonian cycle.

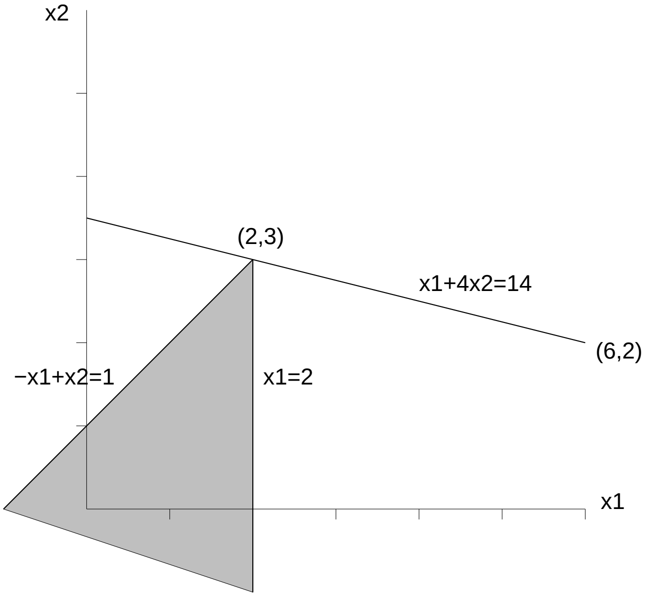
the Hamiltonian Cycle Problem is in $\mathcal{NP}$
 because a Hamiltonian cycle is a succinct certificate for a "yes" answer
the Hamiltonian Cycle Problem $\mathcal{NP}$-complete
 & does not seem to have a succinct certificate for a "no" answer

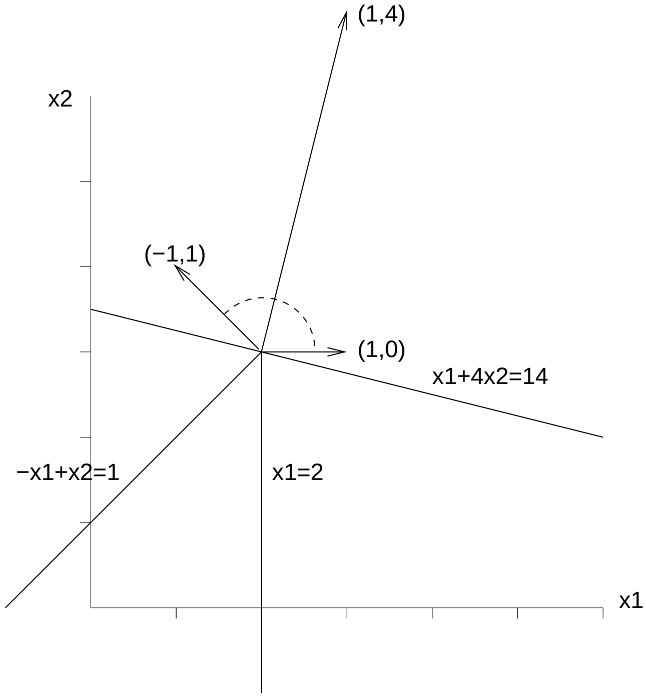
the Disjoint Paths Problem is in $\mathcal{P}$
both "yes" & "no" answers have succinct certificates:
 "yes" answer: the k paths form a succinct certificate
 "no" answer: the $< k$ separating vertices form a succinct certificate

Example

$$\begin{array}{llll} & & \textit{Primal} & \\ \text{maximize } z = & x_1 + 4x_2 & & \\ \text{subject to} & -x_1 \; + x_2 & \le 1 & \\ & x_1 & \le 2 & \end{array} \qquad \begin{array}{llll} & & \textit{Dual} & \\ \text{minimize } w = & y_1 + 2y_2 & & \\ \text{subject to} & -y_1 \; + y_2 & = 1 & \\ & y_1 & = 4 & \\ & y_1, y_2 & \ge 0 & \end{array}$$

optimum primal: $x_1 = 2$, $x_2 = 3$, $z = 14$
optimum dual: $y_1 = 4$, $y_2 = 5$, $w = 14$





recall that the vector (a, b) is normal to the line $ax + by = c$
and points in the direction of increasing $ax + by$
e.g., see Handout#65

the objective is tangent to the feasible region at corner point $(2, 3)$
$\Longrightarrow$ its normal lies between the normals of the 2 constraint lines defining $(2, 3)$
all 3 vectors point away from the feasible region

$\therefore$ the vector of cost coefficients (i.e., $(1, 4)$) is a nonnegative linear combination of
the constraint vectors defining $(2, 3)$ (i.e., $(-1, 1)$ & $(1, 0)$):
$(1, 4) = 4(-1, 1) + 5(1, 0)$

the linear combination is specified by the optimum dual values $y_1 = 4$, $y_2 = 5$!

The General Law

consider this pair of LPs:

	Primal		Dual
maximize $z =$	$\sum_{j=1}^{n} c_j x_j$	minimize $w =$	$\sum_{i=1}^{m} b_i y_i$
subject to	$\sum_{j=1}^{n} a_{ij} x_j \quad \le b_i$	subject to	$\sum_{i=1}^{m} a_{ij} y_i \quad = c_j$
			$y_i \quad \ge 0$

Remark. the primal is the general form of a polyhedron

Notation:
let P be the feasible region of the primal

we use these vectors:

(x_j) denotes the vector $(x_1, \ldots, x_n)$
(c_j) denotes the vector of cost coefficients
$(a_{i\cdot})$ denotes the vector $(a_{i1}, \ldots, a_{in})$
(y_i) denotes the vector of dual values $(y_1, \ldots, y_m)$

suppose the primal optimum is achieved at corner point (x_j^*)
with the objective tangent to P

(x_j^*) is the intersection of n hyperplanes of P
let them be for the first n constraints, with normal vectors $(a_{i\cdot})$, $i = 1, \ldots, n$

(c_j) is a nonnegative linear combination of the n normal vectors $(a_{i\cdot})$, $i = 1, \ldots, n$
i.e., $c_j = \sum_{i=1}^{m} a_{ij} y_i$ where $y_i = 0$ for $i > n$
$\therefore$ (y_i) is dual feasible

it's obvious that (x_j^*) & (y_i) satisfy Complementary Slackness, so (y_i) is optimal

Conclusion: *Suppose an LP has a unique optimum point. The cost vector is a nonnegative linear combination of the constraint vectors that define the optimum corner point. The optimum duals are the mulitipliers in that linear combination.*

Summary:
dual feasibility says (c_j) is a nonnegative linear combination of hyperplane normals
complementary slackness says only hyperplanes defining x^* are used

Primal Problem, a (continuous) knapsack LP:

$$\begin{aligned} \text{maximize}\quad & 9x_1 + 12x_2 + 15x_3 \\ \text{subject to}\quad & x_1 + 2x_2 + 3x_3 \le 5 \\ & x_j \le 1 \quad j = 1,2,3 \\ & x_j \ge 0 \quad j = 1,2,3 \end{aligned}$$

Optimum Solution: $x_1 = x_2 = 1,\ x_3 = 2/3$, objective $z = 31$

Optimum Dictionary

$$\begin{aligned} x_3 &= \frac{2}{3} - \frac{1}{3}s_0 + \frac{1}{3}s_1 + \frac{2}{3}s_2 \\ x_1 &= 1 - s_1 \\ x_2 &= 1 - s_2 \\ s_3 &= \frac{1}{3} + \frac{1}{3}s_0 - \frac{1}{3}s_1 - \frac{2}{3}s_2 \\ \hline z &= 31 - 5s_0 - 4s_1 - 2s_2 \end{aligned}$$

Dual LP:

$$\begin{aligned} \text{minimize}\quad & 5y_0 + y_1 + y_2 + y_3 \\ \text{subject to}\quad & y_0 + y_1 \ge 9 \\ & 2y_0 + y_2 \ge 12 \\ & 3y_0 + y_3 \ge 15 \\ & y_i \ge 0 \quad i = 0, \ldots, 3 \end{aligned}$$

Optimum Dual Solution: $y_0 = 5, y_1 = 4, y_2 = 2, y_3 = 0$, objective $z = 31$

Multiplier Interpretion of Duals

adding $5 \times [x_1 + 2x_2 + 3x_3 \le 5] + 4 \times [x_1 \le 1] + 2 \times [x_2 \le 1]$ shows $9x_1 + 12x_2 + 15x_3 \le 31$
i.e., proof of optimality
obviously we don't use $x_3 \le 1$ in the proof

Complementary Slackness

every x_j positive $\Longrightarrow$ every dual constraint holds with equality
first 3 y_i's positive $\Longrightarrow$ the knapsack constraint & 1st 2 upper bounds hold with equality

Testing Optimality

we verify (x_j) is optimal:
(2): inequality in 3rd upper bound $\Longrightarrow y_3 = 0$

$$(1): \quad \begin{aligned} y_0 + y_1 &= 9 \\ 2y_0 + y_2 &= 12 \\ 3y_0 &= 15 \end{aligned}$$

$\Longrightarrow y_0 = 5, y_2 = 2, y_1 = 4$

(3): holds by definition
(4): holds
$\Longrightarrow (x_j)$ is optimum

Duals are Prices

How valuable is more knapsack capacity?

suppose we increase the size of the knapsack by ϵ
we can add $\epsilon/3$ more pounds of item 3
increasing the value by 5ϵ
so the marginal price of knapsack capacity is 5 $(= y_0)$

How valuable is more of item 3?

obviously 0 $(= y_3)$

How valuable is more of item 1?

suppose ϵ more pounds of item 1 are available
we can add ϵ more pounds of item 1 to the knapsack
assuming we remove $\epsilon/3$ pounds of item 3
this increases the knapsack value by $9\epsilon - 5\epsilon = 4\epsilon$
so the marginal price of item 1 is 4 $(= y_1)$

General Knapsack LPs

Primal:

$$\begin{aligned} \text{maximize} \quad & \sum_{j=1}^{n} v_j x_j \\ \text{subject to} \quad & \sum_{j=1}^{n} w_j x_j \le C \\ & x_j \le 1 \quad j = 1, \ldots, n \\ & x_j \ge 0 \quad j = 1, \ldots, n \end{aligned}$$

Dual:

$$\begin{aligned} \text{minimize} \quad & C y_0 + \sum_{j=1}^{n} y_j \\ \text{subject to} \quad & w_j y_0 + y_j \ge v_j \quad j = 1, \ldots, n \\ & y_j \ge 0 \quad j = 0, \ldots, n \end{aligned}$$

Optimal Solutions

assume the items are indexed by decreasing value per pound, i.e.,
$v_1/w_1 \ge v_2/w_2 \ge \ldots$

optimum solution: using the greedy algorithm, for some s we get
$x_1 = \ldots = x_{s-1} = 1,\ x_{s+1} = \ldots = x_n = 0$

to verify its optimality & compute optimum duals:

(2): $y_{s+1} = \ldots = y_n = 0$ (intuitively clear: they're worthless!)

we can assume $x_s > 0$
now consider 2 cases:

Case 1: $x_s < 1$

(2): $y_s = 0$

(1): equation for x_s gives
$y_0 = v_s/w_s$
equations for x_j, $j < s$ give
$y_j = v_j - w_j(v_s/w_s)$

(4): holds by our indexing

(3): first s equations hold by definition
remaining inequalities say $y_0 \ge v_j/w_j$, true by indexing

Case 2: $x_s = 1$

(1): a system of $s+1$ unknowns $y_j, j = 0, \ldots, s$ & s equations
solution is not unique

but for prices, we know item s is worthless
so we can set $y_s = 0$ and solve as in Case 1

Duals are Prices

our formulas for the duals confirm their interpretation as prices

the dual objective $Cy_0 + \sum_{j=1}^{n} y_j$
computes the value of the knapsack and the items on hand
(i.e., the value of our scarse resources)

the dual constraint $w_j y_0 + y_j \ge v_j$
says the monetary (primal) value of item j is no more than its value computed by price

a vector is a column vector or a row vector, i.e., an $n \times 1$ or $1 \times n$ matrix
so matrix definitions apply to vectors too

Notation

let $\mathbf{x}$ be a row vector & S be an ordered list of indices (of columns)
$\mathbf{x}_S$ is the row vector of columns of $\mathbf{x}$ corresponding to S, ordered as in S
e.g., for $\mathbf{x} = \begin{bmatrix} 5 & 3 & 8 & 1 \end{bmatrix}$, $\mathbf{x}_{(2,1,4)} = \begin{bmatrix} 3 & 5 & 1 \end{bmatrix}$
define $\mathbf{x}_S$ similarly if $\mathbf{x}$ is a column vector
define $\mathbf{A}_S$ similarly if $\mathbf{A}$ is a matrix
where we extract the columns corresponding to S if S is a list of column indices
& similarly for rows

$\mathbf{I}$ is the identity matrix, e.g., $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
the dimension of $\mathbf{I}$ is unspecified and determined by context

similarly $\mathbf{0}$ is the column vector of 0's, e.g. $\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$
with dimension determined by context

Matrix Operations

scalar multiple: for $\mathbf{A} = [a_{ij}]$ an $m \times n$ matrix, t a real number
$t\mathbf{A}$ is an $m \times n$ matrix, $[ta_{ij}]$

matrix sum: for $m \times n$ matrices $\mathbf{A} = [a_{ij}]$, $\mathbf{B} = [b_{ij}]$
$\mathbf{A} + \mathbf{B}$ is an $m \times n$ matrix, $[a_{ij} + b_{ij}]$

matrix product: for $m \times n$ matrix $\mathbf{A} = [a_{ij}]$, $n \times p$ matrix $\mathbf{B} = [b_{jk}]$
$\mathbf{AB}$ is an $m \times p$ matrix with ikth entry $\sum_{j=1}^{n} a_{ij} b_{jk}$

time to multiply two $n \times n$ matrices:
$O(n^3)$ using the definition
$O(n^{2.38})$ using theoretically efficient but impractical methods

in practice much faster than either bound, for sparse matrices–
only store the nonzero elements and their position
only do work on nonzero elements

matrix multiplication is associative, but not necessarily commutative

$\mathbf{AI} = \mathbf{IA} = \mathbf{A}$ for every $n \times n$ matrix $\mathbf{A}$

(see also Handout# 65 for more background on matrices)

Matrix Relations

for $\mathbf{A}$, $\mathbf{B}$ matrices of the same shape,

$\mathbf{A} \le \mathbf{B}$ means $a_{ij} \le b_{ij}$ for all entries

$\mathbf{A} < \mathbf{B}$ means $a_{ij} < b_{ij}$ for all entries

Linear Independence & Nonsingularity

a *linear combination* of vectors $\mathbf{x}^i$ is the sum $\sum_i t_i \mathbf{x}^i$, for some real numbers t_i

if some t_i is nonzero the combination is *nontrivial*

a set of vectors $\mathbf{x}^i$ is *linearly dependent* if some nontrivial linear combination of $\mathbf{x}^i$ equals $\mathbf{0}$

let $\mathbf{A}$ be an $n \times n$ matrix

$\mathbf{A}$ is *singular* $\iff$ the columns of $\mathbf{A}$ are linearly dependent
$\iff$ some nonzero vector $\mathbf{x}$ satisfies $\mathbf{Ax} = \mathbf{0}$
$\iff$ for every column vector $\mathbf{b}$, $\mathbf{Ax} = \mathbf{b}$
has no solution or an infinite number of solutions

$\mathbf{A}$ is *nonsingular* $\iff_0$ the columns of $\mathbf{A}$ are linearly independent
$\iff_1$ for every column vector $\mathbf{b}$, $\mathbf{Ax} = \mathbf{b}$ has exactly one solution
$\iff_2$ $\mathbf{A}$ has an *inverse*, i.e., an $n \times n$ matrix $\mathbf{A}^{-1}$
$\ni \mathbf{AA}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$

Proof.

$_0 \Longrightarrow {}_1$:

≥ 1 solution:

n column vectors in $\mathbf{R}^n$ that are linearly independent *span* $\mathbf{R}^n$,
i.e., any vector is a linear combination of them

≤ 1 solution: $\mathbf{Ax} = \mathbf{Ax}' = \mathbf{b} \Longrightarrow \mathbf{A}(\mathbf{x} - \mathbf{x}') = \mathbf{0} \Longrightarrow \mathbf{x} = \mathbf{x}'$

$_1 \Longrightarrow {}_2$:

construct $\mathbf{A}^{-1}$ column by column so $\mathbf{AA}^{-1} = \mathbf{I}$

to show $\mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$:

$\mathbf{A}(\mathbf{A}^{-1}\mathbf{A}) = \mathbf{A}$, so deduce column by column that $\mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$

$_2 \Longrightarrow {}_0$:

$\mathbf{Ax} = \mathbf{0} \Longrightarrow \mathbf{x} = \mathbf{A}^{-1}\mathbf{Ax} = \mathbf{A}^{-1}\mathbf{0} = \mathbf{0}$ $\square$

consider LP $\mathcal{E}$ of Handout #23
it's solved by the standard simplex in 2 pivots:

Initial Dictionary	x_1 *enters,* x_3 *leaves*	x_2 *enters,* x_4 *leaves*
$x_3 = 1 - x_1$	$x_1 = 1 - x_3$	$x_1 = 1 - x_3$
$x_4 = 2 - x_1 - x_2$	$x_4 = 1 - x_2 + x_3$	$x_2 = 1 + x_3 - x_4$
$z = \quad 3x_1 + x_2$	$z = 3 + x_2 - 3x_3$	$z = 4 - 2x_3 - x_4$
		Optimum Dictionary

Revised Simplex Algorithm for $\mathcal{E}$

in matrix form of $\mathcal{E}$,

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix}, \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \mathbf{c} = \begin{bmatrix} 3 & 1 & 0 & 0 \end{bmatrix}$$

initially $B = (3, 4)$, $\mathbf{x}^*_B = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$

1st Iteration
since we start with the basis of slacks, $\mathbf{B} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, the identity matrix
thus all linear algebra is trivial
this is usually true in general for the first iteration

Entering Variable Step

$$\mathbf{yB} = \mathbf{yI} = \mathbf{y} = \mathbf{c}_B = \begin{bmatrix} 0 & 0 \end{bmatrix}$$

in computing costs, $\mathbf{yA}_{.s} = \mathbf{0}$, so costs are the given costs, as expected
choose x_1 as the entering variable, $c_1 = 3 > 0$

Leaving Variable Step

$$\mathbf{Bd} = \mathbf{d} = \mathbf{A}_{.s} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\mathbf{x}^*_B - t\mathbf{d} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} - t\begin{bmatrix} 1 \\ 1 \end{bmatrix} \geq \mathbf{0}$$

take $t = 1$, x_3 (1st basic variable) leaves

Pivot Step

$$\mathbf{x}^*_B - t\mathbf{d} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} - \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$\mathbf{x}_B^* = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$
$B = (1, 4)$

2nd Iteration

Entering Variable Step

$\mathbf{y}\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 3 & 0 \end{bmatrix}$, $\mathbf{y} = \begin{bmatrix} 3 & 0 \end{bmatrix}$
trying x_2, $1 > \begin{bmatrix} 3 & 0 \end{bmatrix}\begin{bmatrix} 0 \\ 1 \end{bmatrix} = 0$ so it enters

Leaving Variable Step

$\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}\mathbf{d} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, $\mathbf{d} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$
$\begin{bmatrix} 1 \\ 1 \end{bmatrix} - t\begin{bmatrix} 0 \\ 1 \end{bmatrix} \geq \mathbf{0}$
$t = 1$, x_4 (2nd basic variable) leaves

Pivot Step

$\begin{bmatrix} 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$
$\mathbf{x}_B^* = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$
$B = (1, 2)$

3rd Iteration

Entering Variable Step

$\mathbf{y}\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 3 & 1 \end{bmatrix}$, $\mathbf{y} = \begin{bmatrix} 2 & 1 \end{bmatrix}$
all nonbasic costs are nonpositive:

$$\begin{bmatrix} 0 & 0 \end{bmatrix} - \begin{bmatrix} 2 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = -\begin{bmatrix} 2 & 1 \end{bmatrix}$$

$\Longrightarrow$ optimum solution

Approach

a revised simplex iteration solves 2 systems, $\mathbf{yB} = \mathbf{c}_B$, $\mathbf{Bd} = \mathbf{A}_{\cdot s}$
then replaces rth column of $\mathbf{B}$ by $\mathbf{A}_{\cdot s}$
& solves similar systems for this new $\mathbf{B}$

maintaining $\mathbf{B}$ in a factored form makes the systems easy to solve & maintain
also maintains sparsity, numerically stable

$\mathbf{Bd} = \mathbf{A}_{\cdot s} \implies$ the next $\mathbf{B}$ matrix is $\mathbf{BE}$, where $\mathbf{E}$ is an eta matrix with rth column $= \mathbf{d}$

thus $\mathbf{B}_k = \mathbf{B}_\ell \mathbf{E}_{\ell+1} \mathbf{E}_{\ell+2} \ldots \mathbf{E}_{k-1} \mathbf{E}_k \qquad (*)$

where $\mathbf{B}_i =$ the basis after i iterations
$\mathbf{E}_i =$ the eta matrix used in the ith iteration to get $\mathbf{B}_i$
$0 \le \ell \le k$

$(*)$ is the *eta factorization of the basis*

Case 1. The Early Pivots

in $(*)$ take $\ell = 0$, $\mathbf{B}_0 = \mathbf{I}$ (assuming the initial basis is from slacks)

the systems of iteration $(k+1)$, $\mathbf{yB}_k = \mathbf{c}_B$, $\mathbf{B}_k\mathbf{d} = \mathbf{A}_{\cdot s}$,
become $\mathbf{yE}_1 \ldots \mathbf{E}_k = \mathbf{c}_B$, $\mathbf{E}_1 \ldots \mathbf{E}_k\mathbf{d} = \mathbf{A}_{\cdot s}$
solve them as eta systems

this method slows down as k increases
eventually it pays to reduce the size of $(*)$ by *refactoring the basis*:
suppose we've just finished iteration ℓ
extract the current base $\mathbf{B}_\ell$ from $\mathbf{A}$, using the basis heading
compute a triangular factorization for $\mathbf{B}_\ell$,

$$\mathbf{U} = \mathbf{L}_m \mathbf{P}_m \ldots \mathbf{L}_1 \mathbf{P}_1 \mathbf{B}_\ell \qquad (\dagger)$$

added benefit of refactoring: curtails round-off errors

Case 2. *Later Pivots*

let $\mathbf{B}_k$ be the current basis
let $\mathbf{B}_\ell$ be the last basis with a triangular factorization

To Solve $\mathbf{B}_k\mathbf{d} = \mathbf{A}_{\cdot s}$
using $(*)$ this system becomes $\mathbf{B}_\ell\mathbf{E}_{\ell+1}\ldots\mathbf{E}_k\mathbf{d} = \mathbf{A}_{\cdot s}$
using $(\dagger)$ this becomes $\mathbf{U}\mathbf{E}_{\ell+1}\ldots\mathbf{E}_k\mathbf{d} = \mathbf{L}_m\mathbf{P}_m\ldots\mathbf{L}_1\mathbf{P}_1\mathbf{A}_{\cdot s}$

to solve,
1. set $\mathbf{a} = \mathbf{L}_m\mathbf{P}_m\ldots\mathbf{L}_1\mathbf{P}_1\mathbf{A}_{\cdot s}$
   multiply right-to-left, so always work with a column vector
2. solve $\mathbf{U}\mathbf{E}_{\ell+1}\ldots\mathbf{E}_k\mathbf{d} = \mathbf{a}$
   treating $\mathbf{U}$ as a product of etas, $\mathbf{U} = \mathbf{U}_m\ldots\mathbf{U}_1$

this procedure accesses the *eta file*
$\mathbf{P}_1, \mathbf{L}_1, \ldots, \mathbf{P}_m, \mathbf{L}_m, \mathbf{U}_m, \ldots \mathbf{U}_1, \mathbf{E}_{\ell+1}, \ldots, \mathbf{E}_k$
in forward (left-to-right) order

the pivot adds the next eta matrix $\mathbf{E}_{k+1}$ (with eta column $\mathbf{d}$) to the end of the file

To Solve $\mathbf{y}\mathbf{B}_k = \mathbf{c}_B$
using $(*)$ this system becomes $\mathbf{y}\mathbf{B}_\ell\mathbf{E}_{\ell+1}\ldots\mathbf{E}_k = \mathbf{c}_B$
to use $(\dagger)$ write $\mathbf{y} = \mathbf{z}\mathbf{L}_m\mathbf{P}_m\ldots\mathbf{L}_1\mathbf{P}_1$, so $\mathbf{z}\mathbf{U}\mathbf{E}_{\ell+1}\ldots\mathbf{E}_k = \mathbf{c}_B$

to solve,
1. solve $\mathbf{z}\mathbf{U}\mathbf{E}_{\ell+1}\ldots\mathbf{E}_k = \mathbf{c}_B$ (treating $\mathbf{U}$ as a product of etas)
2. $\mathbf{y} = \mathbf{z}\mathbf{L}_m\mathbf{P}_m\ldots\mathbf{L}_1\mathbf{P}_1$
   multiply left-to-right

this accesses the eta file in reverse order
so this method has good locality of reference

obviously the early pivots are a special case of this scheme

other implementation issues: in-core vs. out-of-core; pricing strategies; zero tolerances

Efficiency of Revised Simplex

Chvátal estimates optimum refactoring frequency = every 16 iterations
gives (average # arithmetic operations per iteration) $= 32m + 10n$
versus $mn/4$ for standard simplex (even assuming sparsity)

i.e., revised simplex is better when $(m-40)n \ge 128m$, e.g.,
$n \ge 2m$ (expected in practice) & $m \ge 104$
$m \approx 100$ is a small LP
today's large LPs have thousands or even millions of variables

Principles in Chvátal's Time Estimate

1. the eta columns of $\mathbf{E}_i$ have density between .25 & .5
   in the "steady state", i.e., after the slacks have left the basis
   density is much lower before this

2. solving $\mathbf{B}_k\mathbf{d} = \mathbf{A}_{\cdot s}$ is twice as fast as $\mathbf{y}\mathbf{B}_k = \mathbf{c}_B$

   Argument:
   $\mathbf{c}_B$ is usually dense, but not $\mathbf{A}_{\cdot s}$
   we compute the solution $\mathbf{d}^{i+1}$ to $\mathbf{E}_{i+1}\mathbf{d}^{i+1} = \mathbf{d}^i$
   $\mathbf{d}^i$ is expected to have the density given in #1
   (since it could have been an eta column)

   $\Longrightarrow$ if $\mathbf{E}_{i+1}$ has eta column p, $d^i_p = 0$ with probability $\geq .5$
   but when $d^i_p = 0$, $\mathbf{d}^{i+1} = \mathbf{d}^i$, i.e., no work done for $\mathbf{E}_{i+1}$!

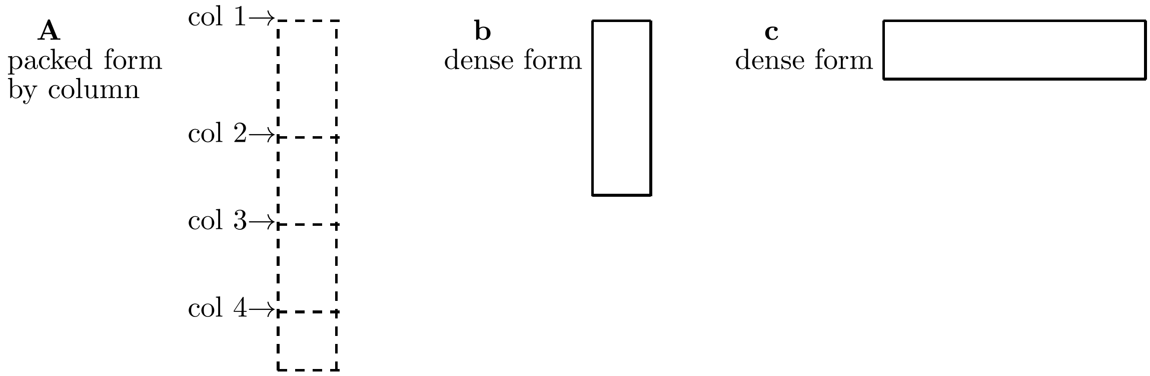
3. in standard simplex, storing the entire dictionary can create core problems
   also the sparse data structure for standard simplex is messy (e.g., Knuth, Vol. I) –
   dictionary must be accessed by row (pivot row) & column (entering column)

Product Form of the Inverse – a commonly-used implementation of revised simplex
maintains $\mathbf{B}_k^{-1} = \mathbf{E}_k\mathbf{E}_{k-1}\ldots\mathbf{E}_1$
where $\mathbf{E}_i$ = eta matrix that specifies the ith pivot

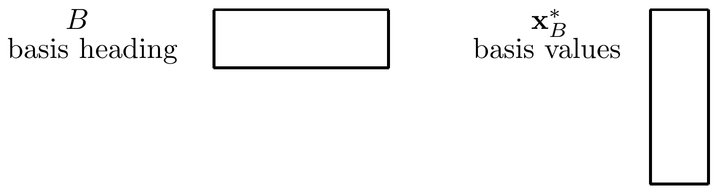
Enhanced Triangular Factorization of the Basis (Chvátal, Ch. 24) –
achieves even greater sparsity, halving the number of nonzeros

Data Structures

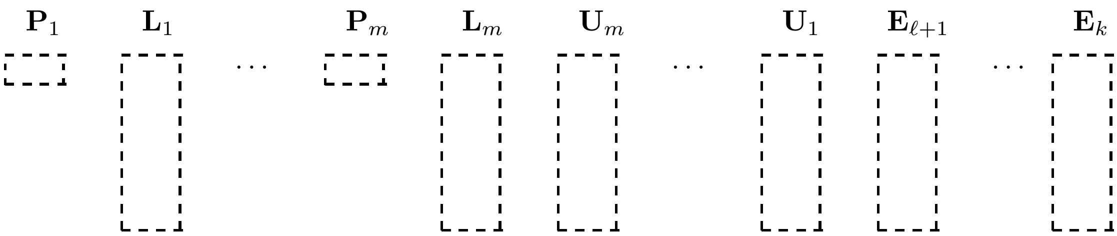
given data:



basis:



eta file:



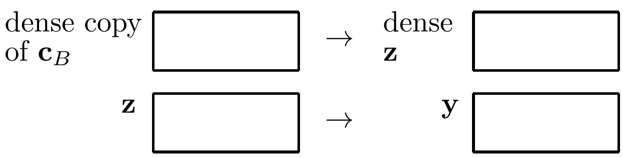
$$\mathbf{L}_m\mathbf{P}_m \dots \mathbf{L}_1\mathbf{P}_1\mathbf{B} = \mathbf{U}\mathbf{E}_{\ell+1} \dots \mathbf{E}_k$$

($\mathbf{B}$ is the current basis)

Entering Variable Step

Solve $\mathbf{yB} = \mathbf{c}_B$:

1. solve $\mathbf{zUE}_{\ell+1} \dots \mathbf{E}_k = \mathbf{c}_B$
2. $\mathbf{y} = \mathbf{zL}_m\mathbf{P}_m \dots \mathbf{L}_1\mathbf{P}_1$



Choose any (nonbasic) $s \ni c_s > \mathbf{yA}_{\cdot s}$

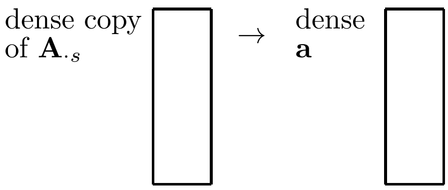
compute $\mathbf{yA}_{\cdot s}$ using dense $\mathbf{y}$, packed $\mathbf{A}_{\cdot s}$

If none exists, stop, B is an optimum basis

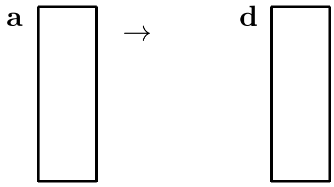
Remark. Steps 1–2 read the eta file backwards, so LP practitioners call them BTRAN ("backward transformation")

Leaving Variable Step
Solve $\mathbf{Bd} = \mathbf{A}_{\cdot s}$:

1. $\mathbf{a} = \mathbf{L}_m\mathbf{P}_m \ldots \mathbf{L}_1\mathbf{P}_1\mathbf{A}_{\cdot s}$



2. solve $\mathbf{U}\mathbf{E}_{\ell+1} \ldots \mathbf{E}_k\mathbf{d} = \mathbf{a}$



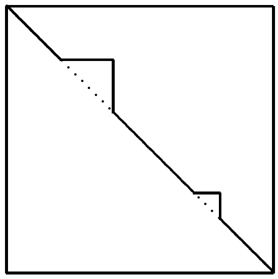
Add packed copy of $\mathbf{E}_{k+1}$ to eta file
Let t be the largest value $\ni$ $\mathbf{x}_B^* - t\mathbf{d} \geq \mathbf{0}$
    dense vectors
If $t = \infty$, stop, the problem is unbounded
Otherwise choose a (basic) r whose component of $\mathbf{x}_B^* - t\mathbf{d}$ is zero

Remark. Steps 1–2 read the eta file forwards & are called FTRAN ("forward transformation")

Pivot Step
In basis heading B replace r by s
$\mathbf{x}_B^* \leftarrow \mathbf{x}_B^* - t\mathbf{d}$
In $\mathbf{x}_B^*$, replace entry for r (now 0) by t

Refactoring Step (done every 20 iterations)
Use B to extract $\mathbf{B} = \mathbf{A}_B$ from packed matrix $\mathbf{A}$
Convert $\mathbf{B}$ to linked list format
Execute Gaussian elimination on $\mathbf{B}$
    for ith pivot, record $\mathbf{P}_i$, $\mathbf{L}_i$ in new eta file
    & record ith row of $\mathbf{U}$ in the packed vectors $\mathbf{U}$.
At end, add the $\mathbf{U}$ vectors to the eta file

Remark.
to achieve a sparser triangular factorization,
we may permute the rows and columns of $\mathbf{B}$ to make it almost lower triangular form,
with a few spikes (Chvátal, p.91–92)



The spikes can create fill-in.

to adjust for permuting columns, do the same permutation on B & $\mathbf{x}_B^*$
to adjust for permuting rows by $\mathbf{P}$, make $\mathbf{P}$ the first matrix of the eta file

Exercise. Verify this works.

we want to solve a system of linear inequalities $\mathcal{I}$,

$$\sum_{j=1}^{n} a_{ij}x_j \le b_i, \quad i = 1, \ldots, m$$

this problem, LI, is equivalent to LP (Exercise of Handout#18)

Fourier (1827) & later Motzkin (1936) proposed a simple method to solve inequality systems:
elimination & back substitution
usually inefficient, but has some applications

Recursive Algorithm to Find a Solution to $\mathcal{I}$

1. rewrite each inequality involving x_1 in the form $x_1 \le u$ or $x_1 \ge \ell$,
   where each u, ℓ is an affine function of $x_2, \ldots, x_n$, $\sum_{j=2}^{n} c_j x_j + d$

2. form $\mathcal{I}'$ from $\mathcal{I}$ by replacing the inequalities involving x_1 by inequalities $\ell \le u$
   ℓ ranges over all lower bounds on x_1, u ranges over all upper bounds on x_1
   $\mathcal{I}'$ is a system on $x_2, \ldots, x_n$

3. delete any redundant inequalities from $\mathcal{I}'$

4. recursively solve $\mathcal{I}'$

5. if $\mathcal{I}'$ is infeasible, so is $\mathcal{I}$
   if $\mathcal{I}'$ is feasible, choose x_1 so (the largest ℓ) $\le x_1 \le$ (the smallest u) $\square$

unfortunately Step 3 is hard, & repeated applications of Step 2 can generate huge systems

but here's an example where Fourier-Motzkin works well:

consider the system $\quad x_i - x_j \le b_{ij} \quad i, j = 1, \ldots, n, \quad i \ne j$

write x_1's inequalities as $\quad x_1 \le x_j + b, \quad x_1 \ge x_k + b$
eliminating x_1 creates inequalities $\quad x_k - x_j \le b$
so the system on $x_2, \ldots, x_n$ has the original form
& eliminating redundancies (simple!) ensures all systems generated have $\le n^2$ inequalities

thus we solve the given system in time $\underline{O(n^3)}$

Remark
the problem of finding shortest paths in a graph has this form
$O(n^3)$ is the best bound for this problem!

Finite Basis Theorem

says $\mathcal{I}$ has "essentially" a finite # of distinct solutions

first we extend the notion of bfs:
$\mathbf{x}$ is a *normal bfs* for $\mathbf{Ax} \le \mathbf{b}$ if for $\mathbf{v} = \mathbf{b} - \mathbf{Ax}$,

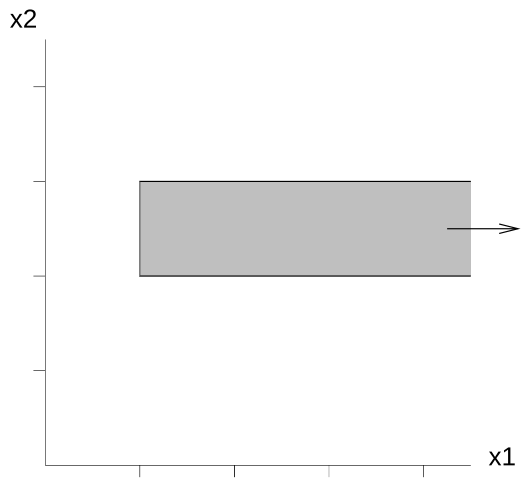
$$\begin{bmatrix} \mathbf{x} \\ \mathbf{v} \end{bmatrix} \text{ is a normal bfs of } \mathbf{Ax} + \mathbf{v} = \mathbf{b},\ \mathbf{v} \ge 0$$

define a *basic feasible direction* of $\mathbf{Ax} \le \mathbf{b}$ in the same way, i.e., introduce slacks

Example.
$x_1 \ge 1$, $2 \le x_2 \le 3$

introducing slacks v_1, v_2, v_3 gives coefficient matrix $\begin{bmatrix} -1 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix}$

bfs's $x_1 = 1, x_2 = 2, v_3 = 1$ & $x_1 = 1, x_2 = 3, v_2 = 1$ are corner points
bfd $x_1 = 1, v_1 = 1$ (basis v_1, v_2, v_3) is direction vector for unboundedness



the vector $\sum_{i=1}^{k} t_i \mathbf{x}^i$ is a

nonnegative combination of $\mathbf{x}^i$, $i = 1, \ldots, k$ if each $t_i \geq 0$

convex combination of $\mathbf{x}^i$, $i = 1, \ldots, k$ if each $t_i \geq 0$ & the t_i's sum to 1

in our example, any feasible point is
a convex combination of the 2 corners
plus a nonnegative combination of the direction vector for unboundedness

this is true in general:

Finite Basis Theorem. *The solutions to* $\mathbf{Ax} \leq \mathbf{b}$ *are precisely the vectors that are convex combinations of* $\mathbf{v}^i$*,* $i = 1, \ldots, M$ *plus nonnegative combinations of* $\mathbf{w}^j$*,* $j = 1, \ldots, N$ *for some finite sets of vectors* $\mathbf{v}^i$*,* $\mathbf{w}^j$*.*

Proof Idea.

the $\mathbf{v}^i$ are the normal bfs'
the $\mathbf{w}^j$ are the bfd's

the argument is based on Farkas' Lemma $\square$

Decomposition Algorithm (Chvátal, Ch.26)

applicable to structured LPs

start with a general form LP $\mathcal{L}$:
maximize $\mathbf{cx}$ subject to $\mathbf{Ax} = \mathbf{b}$, $\ell \le \mathbf{x} \le \mathbf{u}$

break the equality constraints into 2 sets, $\mathbf{A}'\mathbf{x} = \mathbf{b}'$, $\mathbf{A}''\mathbf{x} = \mathbf{b}''$

apply the Finite Basis Theorem to the system $\mathcal{S}$
$\mathbf{A}''\mathbf{x} = \mathbf{b}''$, $\ell \le \mathbf{x} \le \mathbf{u}$
to get that any fs to $\mathcal{S}$ has the form
$\mathbf{x} = \sum_{i=1}^{M} r_i \mathbf{v}^i + \sum_{j=1}^{N} s_j \mathbf{w}^j$
for r_i, s_j nonnegative, $\sum_{i=1}^{M} r_i = 1$, and $\mathbf{v}^i, \mathbf{w}^j$ as above

rewrite $\mathcal{L}$ using the equation for $\mathbf{x}$ to get the "master problem" $\mathcal{M}$:
maximize $\mathbf{c_r r} + \mathbf{c_s s}$ subject to $\mathbf{A_r r} + \mathbf{A_s s} = \mathbf{b}'$, $\sum_{i=1}^{M} r_i = 1$, $r_i, s_j \ge 0$
for vectors $\mathbf{c_r}, \mathbf{c_s}$ & matrices $\mathbf{A_r}, \mathbf{A_s}$ derived from $\mathbf{c}$ & $\mathbf{A}'$ respectively

since M & N are huge, we don't work with $\mathcal{M}$ explicitly – instead solve $\mathcal{M}$ by column generation:

each Entering Variable Step solves the auxiliary problem $\mathcal{A}$:
maximize $\mathbf{c} - \mathbf{yA}'$ subject to $\mathbf{A}''\mathbf{x} = \mathbf{b}''$, $\ell \le \mathbf{x} \le \mathbf{u}$
where $\mathbf{y}$ is the vector of dual values for $\mathcal{M}$, with its last component dropped

the solution to $\mathcal{A}$ will be either
- a normal bfs (i.e., a $\mathbf{v}^i$) which can enter $\mathcal{M}$'s basis, or
- a basic feasible direction of an unbounded solution (a $\mathbf{w}^j$) which can enter $\mathcal{M}$'s basis, or
- a declaration of optimality

the decomposition algorithm works well when we can choose $\mathbf{A}', \mathbf{A}''$ so
$\mathbf{A}'$ has few constraints, and either

$\mathbf{A}''$ can be solved fast, e.g., a network problem, or
$\mathbf{A}''$ breaks up into smaller independent LPs, so we can solve small auxiliary problems

consider a standard form LP & its dual:

Primal Problem $\mathcal{P}$	*Dual Problem $\mathcal{D}$*
maximize $z = \mathbf{cx}$	minimize $\mathbf{yb}$
subject to $\mathbf{Ax} \le \mathbf{b}$ $\mathbf{x} \ge \mathbf{0}$	subject to $\mathbf{yA} \ge \mathbf{c}$ $\mathbf{y} \ge \mathbf{0}$

Theorem 1. *The standard dual simplex algorithm for $\mathcal{P}$ amounts to executing the standard simplex algorithm on the dual problem $\mathcal{D}$.*

Theorem 2. *For a standard form LP $\mathcal{P}$, there is a 1-1 correspondence between primal dictionaries (for $\mathcal{P}$) & dual dictionaries (for $\mathcal{D}$) such that*

(i) B is a primal basis $\Longleftrightarrow$ N is a dual basis
(ii) any row in B's dictionary is the negative of a column in N's dictionary:

primal dictionary

$$\begin{array}{ll} x_i = \overline{b}_i - \sum_{j\in N} \overline{a}_{ij} x_j, & i \in B \\ \hline z = \overline{z} + \sum_{j\in N} \overline{c}_j x_j & \end{array}$$

dual dictionary

$$\begin{array}{ll} y_j = -\overline{c}_j + \sum_{i\in B} \overline{a}_{ij} y_i, & j \in N \\ \hline -w = -\overline{z} - \sum_{i\in B} \overline{b}_i y_i & \end{array}$$

Proof of Theorem 1:
show that after each pivot
the 2 simplex algorithms have dictionaries corresponding as in (ii)

argument is straightforward
e.g., dual simplex's minimum ratio test is
 minimize c_s/a_{rs}, $a_{rs} < 0$
standard simplex's minimum ratio test on the corresponding dual dictionary is
 minimize $-c_s/-a_{rs}$, $-a_{rs} > 0$ $\square$

Proof of Theorem 2:

index the primal constraints and variables as follows:
 C = the set of primal constraints ($|C| = m$)
 D = the set of primal "decision" variables (i.e., the given variables; $|D| = n$)

Proof of (i):

primal constraints after introducing slacks:

$$\begin{bmatrix} \mathbf{I} & \mathbf{A} \end{bmatrix} \begin{bmatrix} \mathbf{x}_C \\ \mathbf{x}_D \end{bmatrix} = \mathbf{b}$$

define $\mathbf{P} = \begin{bmatrix} \mathbf{I} & \mathbf{A} \end{bmatrix}$
$\mathbf{x}_C$ consists of m slack variables indexed by C
$\mathbf{x}_D$ consists of n decision variables indexed by D

dual constraints after introducing slacks:

$$\begin{bmatrix} \mathbf{y}_C & \mathbf{y}_D \end{bmatrix} \begin{bmatrix} \mathbf{A} \\ -\mathbf{I} \end{bmatrix} = \mathbf{c}$$

define $\mathbf{Q} = \begin{bmatrix} \mathbf{A} \\ -\mathbf{I} \end{bmatrix}$
$\mathbf{y}_C$: m decision variables
$\mathbf{y}_D$: n slack variables

in (i), B is a set of m indices of $C \cup D$, N is the complementary set of n indices
for simplicity let B consist of
the first k indices of D and last $m-k$ indices of C

denote intersections by dropping the $\cap$ sign –
e.g., BC denotes all indices in both B & C

we write $\mathbf{P}$ with its rows and columns labelled by the corresponding indices:

$$\mathbf{P} = \begin{array}{c} \begin{array}{cccc} \text{NC} & \text{BC} & \text{BD} & \text{ND} \end{array} \\ \begin{bmatrix} \mathbf{I}_k & \mathbf{0} & \mathbf{B} & \mathbf{Y} \\ \mathbf{0} & \mathbf{I}_{m-k} & \mathbf{X} & \mathbf{Z} \end{bmatrix} \end{array} \begin{array}{c} \\ \text{NC} \\ \text{BC} \end{array}$$

B is a primal basis $\iff$ the columns of BC & BD are linearly independent
$\iff$ $\mathbf{B}$ is nonsingular

we write $\mathbf{Q}$ with its rows and columns labelled by the corresponding indices:

$$\mathbf{Q} = \begin{array}{c} \begin{array}{cc} \text{BD} & \text{ND} \end{array} \\ \begin{bmatrix} \mathbf{B} & \mathbf{Y} \\ \mathbf{X} & \mathbf{Z} \\ -\mathbf{I}_k & \mathbf{0} \\ \mathbf{0} & -\mathbf{I}_{n-k} \end{bmatrix} \end{array} \begin{array}{c} \\ \text{NC} \\ \text{BC} \\ \text{BD} \\ \text{ND} \end{array}$$

N is a dual basis $\iff$ the rows of NC & ND are linearly independent
$\iff$ $\mathbf{B}$ is nonsingular $\quad\square$

Proof of (*ii*):

the primal dictionary for basis B is

$$\begin{array}{l} \mathbf{x}_B = \mathbf{P}_B^{-1}\mathbf{b} - \mathbf{P}_B^{-1}\mathbf{P}_N\mathbf{x}_N \\ \hline z = \mathbf{c}_B\mathbf{P}_B^{-1}\mathbf{b} + (\mathbf{c}_N - \mathbf{c}_B\mathbf{P}_B^{-1}\mathbf{P}_N)\mathbf{x}_N \end{array}$$

using our expression for $\mathbf{P}$ we have

$$\mathbf{P}_B = \begin{bmatrix} \mathbf{0} & \mathbf{B} \\ \mathbf{I}_{m-k} & \mathbf{X} \end{bmatrix}, \quad \mathbf{P}_B^{-1} = \begin{bmatrix} -\mathbf{X}\mathbf{B}^{-1} & \mathbf{I}_{m-k} \\ \mathbf{B}^{-1} & \mathbf{0} \end{bmatrix}$$

remembering the dual constraints are $\mathbf{y}\mathbf{Q} = \mathbf{c}$
we derive the dual dictionary for basis N (with nonbasic variables B):
let $\mathbf{Q}_N$ denote the matrix $\mathbf{Q}$ keeping only the rows of N, & similarly for $\mathbf{Q}_B$

$$\begin{array}{l} \mathbf{y}_N = \mathbf{c}\mathbf{Q}_N^{-1} - \mathbf{y}_B\mathbf{Q}_B\mathbf{Q}_N^{-1} \\ \hline z = -\mathbf{c}\mathbf{Q}_N^{-1}\mathbf{b}_N + \mathbf{y}_B(\mathbf{Q}_B\mathbf{Q}_N^{-1}\mathbf{b}_N - \mathbf{b}_B) \end{array}$$

using our expression for $\mathbf{Q}$ we have

$$\mathbf{Q}_N = \begin{bmatrix} \mathbf{B} & \mathbf{Y} \\ \mathbf{0} & -\mathbf{I}_{n-k} \end{bmatrix}, \quad \mathbf{Q}_N^{-1} = \begin{bmatrix} \mathbf{B}^{-1} & \mathbf{B}^{-1}\mathbf{Y} \\ \mathbf{0} & -\mathbf{I}_{n-k} \end{bmatrix}$$

1. now we check the terms $\overline{a}_{ij}$ in the 2 dictionaries (defined in (*ii*)) correspond:
in the primal dictionary these terms are $\mathbf{P}_B^{-1}\mathbf{P}_N$
which equal $\begin{bmatrix} -\mathbf{X}\mathbf{B}^{-1} & \mathbf{I}_{m-k} \\ \mathbf{B}^{-1} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{I}_k & \mathbf{Y} \\ \mathbf{0} & \mathbf{Z} \end{bmatrix}$

in the dual dictionary these terms are $\mathbf{Q}_B\mathbf{Q}_N^{-1}$
which equal $\begin{bmatrix} \mathbf{X} & \mathbf{Z} \\ -\mathbf{I}_k & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{B}^{-1} & \mathbf{B}^{-1}\mathbf{Y} \\ \mathbf{0} & -\mathbf{I}_{n-k} \end{bmatrix}$
the 2 products are negatives of each other, as desired

2. now we check the objective values are negatives of each other:
the primal objective value is $\mathbf{c}_B\mathbf{P}_B^{-1}\mathbf{b}$
$\mathbf{c}_B = \begin{bmatrix} \mathbf{0}_{m-k} & \mathbf{c}_{BD} \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} \mathbf{b}_{NC} \\ \mathbf{b}_{BC} \end{bmatrix}$
objective value $= \begin{bmatrix} \mathbf{c}_{BD}\mathbf{B}^{-1} & \mathbf{0}_{m-k} \end{bmatrix} \mathbf{b} = \mathbf{c}_{BD}\mathbf{B}^{-1}\mathbf{b}_{NC}$

the dual objective value is $-\mathbf{c}\mathbf{Q}_N^{-1}\mathbf{b}_N$:
$\mathbf{c} = \begin{bmatrix} \mathbf{c}_{BD} & \mathbf{c}_{ND} \end{bmatrix}, \quad \mathbf{b}_N = \begin{bmatrix} \mathbf{b}_{NC} \\ \mathbf{0}_{n-k} \end{bmatrix}$
objective value $= -\mathbf{c}\begin{bmatrix} \mathbf{B}^{-1}\mathbf{b}_{NC} \\ \mathbf{0}_{n-k} \end{bmatrix} = -\mathbf{c}_{BD}\mathbf{B}^{-1}\mathbf{b}_{NC}$, negative of primal

3. similar calculations show
primal cost coefficients are negatives of dual r.h.s. coefficients
dual cost coefficients are negatives of primal r.h.s. coefficients $\square$

a polyhedron $\mathbf{Ax} \le \mathbf{b}$ is *rational* if every entry in $\mathbf{A}$ & $\mathbf{b}$ is rational

a rational polyhedron P is *integral* $\iff$ it is the convex hull of its integral points
$\iff$ every (minimal) face of P has an integral vector
$\iff$ every LP max $\mathbf{cx}$ st $\mathbf{x} \in P$ with an optimum point has an integral optimum point
so for integral polyhedra, ILP reduces to LP

Example Application of Integral Polyhedra

a graph is *regular* if every vertex has the same degree
a matching is *perfect* if every vertex is on a matched edge

Theorem. *Any regular bipartite graph has a perfect matching.*

Proof.
take a bipartite graph where every vertex has degree d
let $\mathbf{A}$ be the node-arc incidence matrix
consider the polyhedron $\mathbf{Ax} = \mathbf{1}$, $\mathbf{x} \ge \mathbf{0}$ – we'll see in Theorem 2 that it's integral
the polyhedron has a feasible point: set $x_{ij} = 1/d$ for every edge ij
so there's an integral feasible point, i.e., a perfect matching $\square$

Total Unimodularity

this property, of $\mathbf{A}$ alone, makes a polyhedron integral
a matrix $\mathbf{A}$ is *totally unimodular* if every square submatrix has determinant $0, \pm 1$

Examples of Totally Unimodular Matrices

1. the node-arc incidence matrix of a digraph

   Proof Sketch:
   let $\mathbf{B}$ be a square submatrix of the incidence matrix
   induct on the size of $\mathbf{B}$

   Case 1: every column of $\mathbf{B}$ contains both a $+1$ & a -1
   the rows of $\mathbf{B}$ sum to $\mathbf{0}$, so $\det(\mathbf{B}) = 0$

   Case 2: some column of $\mathbf{B}$ contains only 1 nonzero
   expand $\det(\mathbf{B})$ by this column and use inductive hypothesis $\square$

2. the node-arc incidence matrix of a bipartite graph
   similar proof

3. interval matrices – 0,1 matrices where each column has all its 1's consecutive

Proof Idea:
proceed as above if row 1 contains ≤ 1 positive entry
otherwise, subtract the shorter column from the longer to reduce the total number of 1's

4. an amazing theorem of Seymour characterizes the totally unimodular matrices as being built up from "network matrices" and 2 exceptional 5×5 matrices, using 9 types of operations

Theorem 1. $\mathbf{A}$ *totally unimodular* $\Longrightarrow$
for every integral vector $\mathbf{b}$*, the polyhedron* $\mathbf{Ax} \le \mathbf{b}$ *is integral.*

Theorem 2. *Let* $\mathbf{A}$ *be integral.* $\mathbf{A}$ *totally unimodular*
$\Longleftrightarrow$ *for every integral* $\mathbf{b}$*, the polyhedron* $\mathbf{Ax} \le \mathbf{b}$*,* $\mathbf{x} \ge \mathbf{0}$ *is integral*
$\Longleftrightarrow$ *for every integral* $\mathbf{a}$*,* $\mathbf{b}$*,* $\mathbf{c}$*,* $\mathbf{d}$*, the polyhedron* $\mathbf{a} \le \mathbf{Ax} \le \mathbf{b}$*,* $\mathbf{c} \le \mathbf{x} \le \mathbf{d}$ *is integral*

can also allow components of these vectors to be $\pm\infty$

Proof Idea (this is the basic idea of total unimodularity):
suppose $\mathbf{A}$ is totally unimodular & $\mathbf{b}$ is integral
we'll show the polyhedron $\mathbf{Ax} \le \mathbf{b}$, $\mathbf{x} \ge \mathbf{0}$ is integral

for any basis $\mathbf{B}$, the basic variables have values $\mathbf{B}^{-1}\mathbf{b}$
$\mathbf{B}$ has determinant ± 1, by total unimodularity
note that some columns of $\mathbf{B}$ can be slacks
so $\mathbf{B}^{-1}$ is an integral matrix $\square$

Theorem 2 gives the Transhipment Integrality Theorem
(even with lower and upper bounds on flow)

Theorem 3. *Let* $\mathbf{A}$ *be integral.* $\mathbf{A}$ *totally unimodular* $\Longleftrightarrow$
for every integral $\mathbf{b}$ & $\mathbf{c}$ *where the primal-dual LPs*
max $\mathbf{cx}$ st $\mathbf{Ax} \le \mathbf{b}$, $\mathbf{x} \ge \mathbf{0}$
min $\mathbf{yb}$ st $\mathbf{yA} \ge \mathbf{c}$, $\mathbf{y} \ge \mathbf{0}$
both have an optimum, both LPs have an integral optimum point.

Theorem 3 contains many combinatorial facts, e.g.:
let $\mathbf{A}$ be the incidence matrix of a bipartite graph
let $\mathbf{b}$, $\mathbf{c}$ be vectors of all 1's

König-Egerváry Theorem. *In a bipartite graph,*
the maximum cardinality of a matching equals the minimum cardinality of a vertex cover.

Total Dual Integrality

this property makes a polyhedron integral, but involves $\mathbf{A}$, $\mathbf{b}$, and every $\mathbf{c}$

Illustrative Example: Fulkerson's Arborescence Theorem

take a digraph with nonnegative integral edge lengths $\ell(e)$ & a distinguished vertex r
an *r-arborescence* is a directed spanning tree rooted at vertex r
    all edges are directed away from r
an *r-cut* is a set of vertices not containing r
an *r-cut packing* is a collection of r-cuts, with repetitions allowed,
    such that each edge e enters $\leq \ell(e)$ cuts
    its *size* is the number of sets

Theorem. *For any digraph, ℓ & r,*
the minimum total length of an r-arborescence equals the maximum size of an r-cut packing.

let $\mathbf{C}$ be the r-cut-edge incidence matrix
consider the primal-dual pair,
    max $\mathbf{y1}$ st $\mathbf{yC} \leq \ell$, $\mathbf{y} \geq \mathbf{0}$
    min $\ell\mathbf{x}$ st $\mathbf{Cx} \geq \mathbf{1}$, $\mathbf{x} \geq \mathbf{0}$

it's not hard to see that if both LPs have integral optima, Fulkerson's Theorem holds

it's easy to see that $\mathbf{C}$ is not totally unimodular–
3 edges ra, rb, rc with r-cuts $\{a,b\}, \{a,c\}, \{b,c\}$ give this submatrix with determinant -2:

$$\begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

we'll get Fulkerson's Theorem using the TDI property

take a rational polyhedron $\mathbf{Ax} \leq \mathbf{b}$
consider the primal-dual pair of LPs,
    max $\mathbf{cx}$ st $\mathbf{Ax} \leq \mathbf{b}$
    min $\mathbf{yb}$ st $\mathbf{yA} = \mathbf{c}$, $\mathbf{y} \geq \mathbf{0}$
$\mathbf{Ax} \leq \mathbf{b}$ is *totally dual integral (TDI)* if for every integral $\mathbf{c}$ where the dual has an optimum,
    the dual has an integral optimum point

Theorem 4. $\mathbf{Ax} \leq \mathbf{b}$ *TDI with* $\mathbf{b}$ *integral* $\Longrightarrow$ $\mathbf{Ax} \leq \mathbf{b}$ *is integral.*

returning to Fulkerson:
it can be proved that the system $\mathbf{Cx} \geq \mathbf{1}$, $\mathbf{x} \geq \mathbf{0}$ is TDI, so it is an integral polyhedron
the definition of TDI shows the dual has an integral optimum

so both LPs have integral optima, i.e., Fulkerson's Arborescence Theorem is true

Initialization

in general we need a Phase 1 procedure for initialization
since a transshipment problem can be infeasible (e.g., no source-sink path)
the following Phase 1 procedure sometimes even speeds up Phase 2
(by breaking the network into smaller pieces)

a simple solution vector $\mathbf{x}$ is where 1 node w transships all goods:
all sources i send their goods to w, along edge iw if $i \neq w$
all sinks receive all their demand from w, along edge wi if $i \neq w$
this is feasible (even if w is a source or sink) if all the above edges exist in G
(since satisfying the constraints for all vertices except w implies satisfying w's constraint too)

in general, add every missing edge iw or wi as an *artificial edge*
then run a Phase 1 problem with objective $t = \sum\{x_{wi}, x_{iw} : wi\ (iw) \text{ artificial}\}$

there are 3 possibilities when Phase 1 halts with optimum objective t^*:

1. $t^* > 0$: the given transshipment problem is infeasible
2. $t^* = 0$ & no artificial edge is in the basis: proceed to Phase 2
3. $t^* = 0$ & an artificial edge is in the basis

we now show that in Case 3, the given problem decomposes into smaller subproblems

graph terminology:
V denotes the set of all vertices

for $S \subseteq V$, edge ij *enters* S if $i \notin S$, $j \in S$
ij *leaves* S if $i \in S$, $j \notin S$

Lemma 1. *Let S be a set of vertices where*
(a) no edge of G enters S;
(b) the total net demand in S ($\sum_{i \in S} b_i$) equals 0.
Then any feasible solution (of the given transshipment problem)
has $x_e = 0$ for every edge e leaving S.

Remark. (a) + the Lemma's conclusion show we can solve this network
by finding an optimum solution on S and an optimum solution on $V - S$.

Proof.
(a) shows the demand in S must be satisfied by sources in S
(b) shows this exhausts all the supply in S, i.e., no goods can be shipped out $\square$

let T be the optimum basis from Phase 1

as we traverse an edge from tail to head, y (from Phase 1) increases by ≤ 1
more precisely every edge is oriented like this:

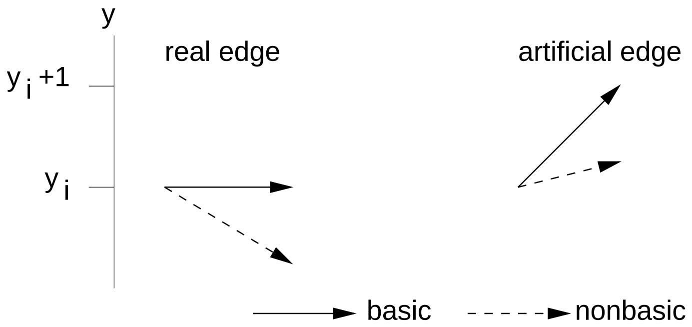


Fig.1. y stays the same on an edge of G in T. It doesn't increase on an edge of $G-T$. Artificial edges increase by ≤ 1.

take any artificial edge $uv \in T$
let $S = \{i : y_i > y_u\}$

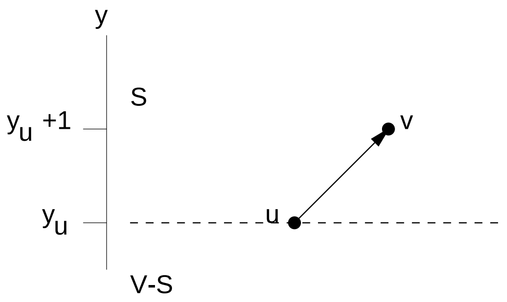


Fig.1 implies
$v \in S$
no edge of G enters S
no edge of G leaving S is in T

thus no goods enter or leave S
this implies the total net demand in S equals 0
now Lemma 1 applies, so the network decomposes into 2 smaller networks

Cycling

the network simplex never cycles, in practice
but Chvátal (p.303) gives a simple example of cycling
it's easy to avoid cycling, as follows

suppose we always use the top-down procedure for computing $\mathbf{y}$ (fixing $y_r = 0$)
it's easy to see a pivot updates $\mathbf{y}$ as follows:

Fact. *Suppose we pivot ij into the basis. Let $\overline{c}_{ij} = c_{ij} + y_i - y_j$. Let d be the deeper end of ij in the new basis. Then $\mathbf{y}$ changes only on descendants of d. In fact*
$d = j$ ($d = i$) $\Longrightarrow$ *every descendant w of d has y_w increase by $\overline{c}_{ij}$ ($-\overline{c}_{ij}$).* $\square$

consider a sequence of consecutive degenerate pivots
each entering edge ij is chosen so $\bar{c}_{ij} < 0$

assume the deeper vertex of ij is always j
the Fact shows $\sum_{k=1}^{n} y_k$ always decreases
this implies we can't return to the starting tree T (since T determines $\sum_{k=1}^{n} y_k$ uniquely)

so it suffices to give a rule that keeps every edge $e \in T$ with $x_e = 0$ directed away from the root
Cunningham's Rule does it, as follows:

suppose 2 or more edge can be chosen to leave the basis
let ij be the entering edge
let a be the nearest common ancestor of i and j in T
traverse the cycle of ij in the direction of ij, starting at a
choose the first edge found that can leave the basis

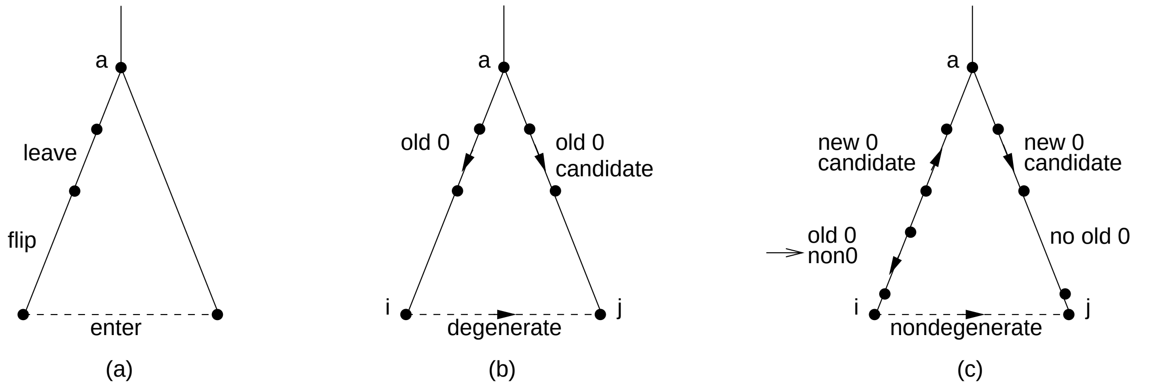


Fig.2. Understanding Cunningham's Rule. "0 edge" means $x_e = 0$.
(a) Edges between the leaving and entering edges flip orientation in a pivot.
(b) Degenerate pivot: the first old 0 edge following j leaves.
(c) Nondegenerate pivot: the first new 0 edge following a leaves.

Implementing Network Simplex Efficiently (Chvátal, 311–317)
tree data structures can be used to speed up the processing of T

using the Fact, $\mathbf{y}$ can be updated by visiting only the descendants of d

a preorder list of T is used to find the descendants of d
Fig.2(a) shows we can update the preorder list in a pivot by working only along the path that flips

Transportation Problem

special case of transshipment:
  no transshipment nodes
  every edge goes from a source to a sink

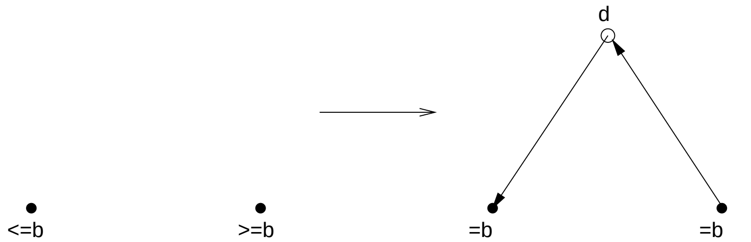
the setting for Hitchcock's early version of network simplex
assignment problem is a special case

we can extend the transshipment problem in 2 ways:

Bounded Sources & Sinks

generalize from demands $= b$ to demands $\geq, \leq b$:

to model such demands add a dummy node d
  route excess goods to d and satisfy shortfalls of goods from d:

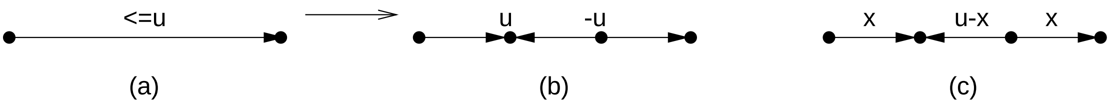


d has demand $-\sum_i b_i$, where the sum is over all real vertices

Bounded Edges

edges usually have a "capacity", i.e., capacity u_{ij} means $x_{ij} \leq u_{ij}$

an edge e with capacity u can be modelled by placing 2 nodes on e:



The capacitated edge of (a) is modelled by (b).
(c) shows how x units flow through the edge. We must have $x \leq u$.

when all edges have capacities (possibly infinite), we have the minimum cost flow problem

Exercise. Sometimes edges have "lower bounds", i.e., lower bound ℓ_{ij} means $x_{ij} \geq \ell_{ij}$. Show how a lower bound can be modelled by decreasing the demand at j & increasing it at i.

Chvátal Ch.21 treats these "upper-bounded transhipment problems" directly, without enlarging the network.

Max Flow Problem (Chvátal Ch.22)

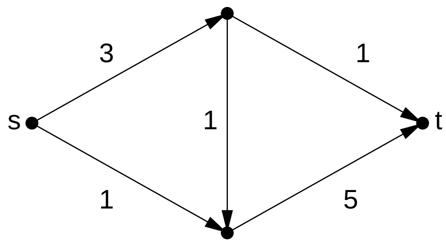
the given graph has 1 source s, with unbounded supply, & 1 sink t, with unbounded demand
each edge has a capacity
the goal is to ship as many units as possible from s to t
  i.e., each edge from s costs -1, all other edges cost 0

Strong Duality has this interpretation:
a *cut* is a set S of vertices that contains s but not t
  the *capacity* of a cut is $\sum_{i \in S, j \notin S} u_{ij}$

obviously the value of any flow is at most the capacity of any cut. Strong Duality says

Max-Flow Min-Cut Theorem. *The maximum value of a flow equals the minimum capacity of a cut.*

for proof see Chvátal p.371



Flow network with capacities.
The max flow & min cut are both 3.

positive definite matrices behave very much like positive numbers
let $\mathbf{A}$ be an $n \times n$ symmetric matrix

$\mathbf{A}$ is *positive definite* $\iff$ (1) every vector $\mathbf{x} \neq \mathbf{0}$ has $\mathbf{x}^T\mathbf{A}\mathbf{x} > 0$
$\iff$ (2) every eigenvalue of $\mathbf{A}$ is positive
$\iff$ (3) $\mathbf{A}$ can be written $\mathbf{B}^T\mathbf{B}$ for some nonsingular matrix $\mathbf{B}$

((3) is like saying every positive number has a nonzero square root)

Example. $\mathbf{A} = \begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix}$ is PD & satisfies (1)–(3):

(1) $2x_1^2 - 2x_1x_2 + x_2^2 = x_1^2 + (x_1 - x_2)^2 > 0$ for $(x_1, x_2) \neq (0, 0)$

(2) $\mathbf{A}\begin{bmatrix} 2 \\ 1 \mp \sqrt{5} \end{bmatrix} = \begin{bmatrix} 3 \pm \sqrt{5} \\ -1 \mp \sqrt{5} \end{bmatrix} \implies$ the eigenvalues are $(3 \mp \sqrt{5})/2$

(3) $\mathbf{A} = \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix}$

Proof.

(1)$\implies$ (2):
suppose $\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$
then $\mathbf{x}^T\mathbf{A}\mathbf{x} = \mathbf{x}^T\lambda\mathbf{x} = \lambda\|\mathbf{x}\|^2 > 0 \implies \lambda > 0$

(2)$\implies$ (3):
$\mathbf{A}$ symmetric $\implies$ it has n orthonormal eigenvectors, say $\mathbf{x}_i, i = 1, \ldots, n$
form $n \times n$ matrix $\mathbf{Q}$, with ith column $\mathbf{x}_i$
form diagonal matrix Λ with ith diagonal entry λ_i, eigenvalue of $\mathbf{x}_i$.
then $\mathbf{AQ} = \mathbf{Q}\Lambda$, $\mathbf{A} = \mathbf{Q}\Lambda\mathbf{Q}^T$
since $\mathbf{Q}^{-1} = \mathbf{Q}^T$

since each eigenvalue is positive, we can write $\Lambda = \mathbf{DD}$
this gives $\mathbf{A} = \mathbf{QD}(\mathbf{QD})^T$

(3)$\implies$ (1):
$\mathbf{x}^T\mathbf{A}\mathbf{x} = \mathbf{x}^T\mathbf{B}^T\mathbf{B}\mathbf{x} = (\mathbf{B}\mathbf{x})^T\mathbf{B}\mathbf{x} = \|\mathbf{B}\mathbf{x}\|^2 > 0$ □

note the factorization $\mathbf{A} = \mathbf{Q}\Lambda\mathbf{Q}^T$ can be computed in polynomial time

2 More Properties of Positive Definite Matrices

1. $\mathbf{A}$ positive definite $\Longrightarrow$ the curve $\mathbf{x}^T\mathbf{A}\mathbf{x} = 1$ defines an ellipsoid, i.e., an n-dimensional ellipse
   Proof.
   using $\mathbf{A} = \mathbf{Q}\Lambda\mathbf{Q}^T$ & substituting $\mathbf{y} = \mathbf{Q}^T\mathbf{x}$, the curve is
   $\mathbf{x}^T\mathbf{Q}\Lambda\mathbf{Q}^T\mathbf{x} = \mathbf{y}^T\Lambda\mathbf{y} = 1$
   the latter equation is $\sum \lambda_i y_i^2 = 1$, an ellipsoid since all eigenvalues are positive
   since $\mathbf{x} = \mathbf{Q}\mathbf{y}$, we rotate the ellipsoid, each axis going into an eigenvector of $\mathbf{A}$ $\square$

2. let $f : \mathbf{R}^n \to \mathbf{R}$, & at some point $\mathbf{x}$,
   If $\nabla f = (\partial f/\partial x_i)$ vanishes & the Hessian matrix $\mathbf{H} = (\partial^2 f/\partial x_i \partial x_j)$ is positive definite then $\mathbf{x}$ is a local minimum

   Proof idea.
   follows from the Taylor series for f,
   $f(\mathbf{x}) = f(\mathbf{0}) + (\nabla f)^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x} +$ (higher order terms)
   $\mathbf{H}$ positive definite $\Longrightarrow$ $f(\mathbf{x}) > f(\mathbf{0})$ for small $\mathbf{x}$ $\square$

$\mathbf{A}$ is *positive semidefinite* $\Longleftrightarrow$ every vector $\mathbf{x}$ has $\mathbf{x}^T\mathbf{A}\mathbf{x} \ge 0$
$\Longleftrightarrow$ every eigenvalue of $\mathbf{A}$ is nonnegative
$\Longleftrightarrow$ $\mathbf{A}$ can be written $\mathbf{B}^T\mathbf{B}$ for some $n \times n$ matrix $\mathbf{B}$

In keeping with the above intuition we sometimes write $\mathbf{X}$ PSD as $\mathbf{X} \succeq 0$

Linear Algebra

the *transpose* of an $m \times n$ matrix $\mathbf{A}$ is the $n \times m$ matrix $\mathbf{A}^T$ where $\mathbf{A}^T_{ij} = \mathbf{A}_{ji}$
$(\mathbf{AB})^T = \mathbf{B}^T\mathbf{A}^T$

the L_2-*norm* $\|\mathbf{x}\|$ of $\mathbf{x} \in \mathbf{R}^n$ is its length according to Pythagoras, $\sqrt{\sum_{i=1}^n x_i^2}$
a *unit vector* has length 1

the *scalar product* of 2 vectors $\mathbf{x}, \mathbf{y} \in \mathbf{R}^n$ is $\mathbf{x}^T\mathbf{y} = \sum_{i=1}^n x_i y_i$
it equals $\|\mathbf{x}\|\|\mathbf{y}\| \cos$(the angle between $\mathbf{x}$ & $\mathbf{y}$)
Cauchy-Schwartz inequality: $\mathbf{x}^T\mathbf{y} \le \|\mathbf{x}\|\|\mathbf{y}\|$

if $\mathbf{y}$ is a unit vector, the scalar product is the length of the projection of $\mathbf{x}$ onto $\mathbf{y}$
$\mathbf{x}$ & $\mathbf{y}$ are *orthogonal* if their scalar product is 0
2 subspaces are *orthogonal* if every vector in one is orthogonal to every vector in the other

an $m \times n$ matrix $\mathbf{A}$ has 2 associated subspaces of $\mathbf{R}^n$:
the *row space* is the subspace spanned by the rows of $\mathbf{A}$
the *nullspace* is the set of vectors $\mathbf{x}$ with $\mathbf{Ax} = \mathbf{0}$

the row space & nullspace are orthogonal (by definition)
in fact they're *orthogonal complements*:
any vector $\mathbf{x} \in \mathbf{R}^n$ can be written uniquely as $\mathbf{r} + \mathbf{n}$
where $\mathbf{r}$ ($\mathbf{n}$) is in the row space (nullspace)
and is called the *projection* of $\mathbf{x}$ onto the row space (nullspace)

Lemma 1. *If the rows of* $\mathbf{A}$ *are linearly independent, the projection of any vector* $\mathbf{x}$ *onto the row space of* $\mathbf{A}$ *is* $\mathbf{A}^T(\mathbf{AA}^T)^{-1}\mathbf{Ax}$.

Proof.
$\mathbf{AA}^T$ is nonsingular:

$$\mathbf{AA}^T\mathbf{y} = \mathbf{0} \Longrightarrow \|\mathbf{A}^T\mathbf{y}\|^2 = (\mathbf{A}^T\mathbf{y})^T\mathbf{A}^T\mathbf{y} = \mathbf{y}^T\mathbf{AA}^T\mathbf{y} = \mathbf{0} \Longrightarrow \mathbf{A}^T\mathbf{y} = \mathbf{0} \Longrightarrow \mathbf{y} = \mathbf{0}$$

the vector of the lemma is in the row space of $\mathbf{A}$
its difference with $\mathbf{x}$ is in the null space of $\mathbf{A}$:

$$\mathbf{A}(\mathbf{x} - \mathbf{A}^T(\mathbf{AA}^T)^{-1}\mathbf{Ax}) = \mathbf{Ax} - \mathbf{Ax} = \mathbf{0} \quad \square$$

an *affine space* is the set $\mathbf{v} + V$ for some linear subspace V, i.e., all vectors $\mathbf{v} + \mathbf{x}$, $\mathbf{x} \in V$

a *ball* $B(\mathbf{v}, r)$ in $\mathbf{R}^n$ is the set of all vectors within distance r of $\mathbf{v}$

Lemma 2. *Let F be the affine space* $\mathbf{v} + V$. *The minimum of an arbitrary linear cost function* $\mathbf{cx}$ *over* $B(\mathbf{v}, r) \cap F$ *is achieved at* $\mathbf{v} - r\mathbf{u}$, *where* $\mathbf{u}$ *is a unit vector along the projection of* $\mathbf{c}$ *onto* V.

Proof.
take any vector $\mathbf{x} \in B(\mathbf{v}, r) \cap F$
let $\mathbf{c}_P$ be the projection of $\mathbf{c}$ onto V

$\mathbf{c}(\mathbf{v}-r\mathbf{u})-\mathbf{c}\mathbf{x}=\mathbf{c}((\mathbf{v}-r\mathbf{u})-\mathbf{x})=\mathbf{c}_P((\mathbf{v}-r\mathbf{u})-\mathbf{x})$ (since $\mathbf{c}-\mathbf{c}_P$ is orthogonal to V)

to estimate the r.h.s.,
Cauchy-Schwartz shows $\mathbf{c}_P(\mathbf{v}-\mathbf{x}) \le \|\mathbf{c}_P\|\|\mathbf{v}-\mathbf{x}\| \le r\|\mathbf{c}_P\|$
$\mathbf{c}_P(-r\mathbf{u}) = -r\|\mathbf{c}_P\|$
so the r.h.s. is ≤ 0, as desired $\square$

Calculus

logarithms:
for all real $x > -1$, $\ln(1+x) \le x$

Lemma 3. *Let $\mathbf{x} \in \mathbf{R}^n$ be a vector with $\mathbf{x} > \mathbf{0}$ and $\sum_{j=1}^n x_j = n$. Set $\alpha = \|\mathbf{1}-\mathbf{x}\|$ & assume $\alpha < 1$. Then*

$$\ln\left(\prod_{j=1}^n x_j\right) \ge \frac{\alpha^2}{\alpha-1}.$$

Exercise 1. Prove Lemma 3. Start by using the general fact that the geometric mean is at most the arithmetic mean:

For any $n \ge 1$ nonnegative numbers $x_j,\ j=1,\ldots,n$,

$$\left(\prod_{j=1}^n x_j\right)^{1/n} \le \left(\sum_{j=1}^n x_j\right)/n$$

(This inequality is tight when all x_j's are equal. It can be easily derived from Jensen's Inequality below.)

Upperbound $\prod_{j=1}^n 1/x_j$ using the above relation. Then write $\mathbf{y}=\mathbf{1}-\mathbf{x}$ & substitute, getting terms $1/(1-y_j)$. Check that $|y_j|<1$, so those terms can be expanded into a geometric series. Simplify using the values of $\sum_{j=1}^n y_j$, $\sum_{j=1}^n y_j^2$. (Use the latter to estimate all high order terms). At the end take logs, & simplify using the above inequality for $\ln(1+x)$.

Lemma 4. *Let H be the hyperplane $\sum_{i=1}^n x_i = 1$. Let Δ be the subset of H where all coordinates x_i are nonnegative. Let $\mathbf{g}=(1/n,\ldots,1/n)$.*
(i) Any point in Δ is at distance at most $R=\sqrt{(n-1)/n}$ from $\mathbf{g}$.
(ii) Any point of H within distance $r = 1/\sqrt{n(n-1)}$ of $\mathbf{g}$ is in Δ.

note that $(1,0,\ldots,0) \in \Delta_n$ and is at distance R from $\mathbf{g}$, since
$(1-1/n)^2 + (n-1)/n^2 = (n-1)n/n^2 = (n-1)/n = R^2$.
hence (i) shows that the smallest circle circumscribed about Δ_n with center $\mathbf{g}$ has radius R

note that $(1/(n-1),\ldots,1/(n-1),0) \in \Delta_n$ and is at distance r from $\mathbf{g}$, since
$(n-1)(1/(n-1)-1/n)^2 + 1/n^2 = 1/n^2(n-1) + 1/n^2 = 1/n(n-1) = r^2$
hence (ii) shows that the largest circle inscribed in Δ_n with center $\mathbf{g}$ has radius r

Exercise 2. Prove Lemma 4. First observe that the function $(x-1/n)^2$ is concave up. (i) follows from this fact. (ii) follows similarly – the handy principle is known as Jensen's Inequality:

If $f(x)$ is concave up, $\sum_{j=1}^n f(x_j) \ge nf(\sum_{j=1}^n x_j/n)$.

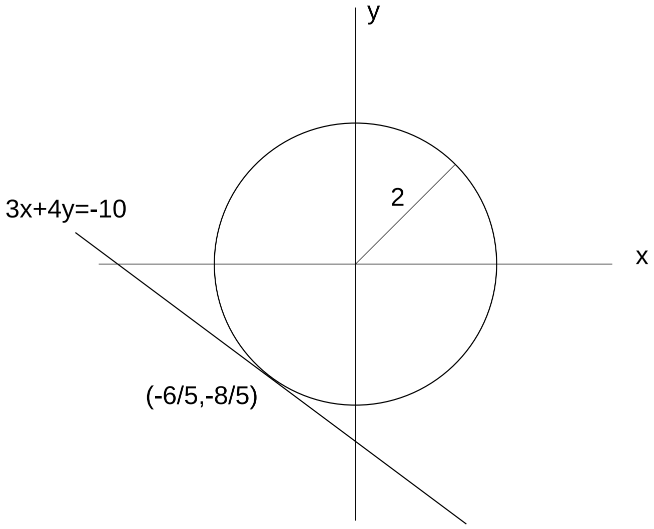
Karmarkar's algorithm advances from a point $\mathbf{p}$ to the next point $\mathbf{p}'$
by a scheme that looks like this:

$$(*)\qquad \begin{tikzcd} \mathbf{p} \arrow[r, "T_p"] & \mathbf{g} \arrow[d, "\text{spherical minimization}"] \\ \mathbf{p}' & \mathbf{s} \arrow[l, "T_p^{-1}"] \end{tikzcd}$$

this handout and the next two explain the basic ideas in $(*)$
then we present the algorithm

Optimizing Over Spheres

it's easy to optimize a linear function over a spherical feasible region
by method of steepest descent



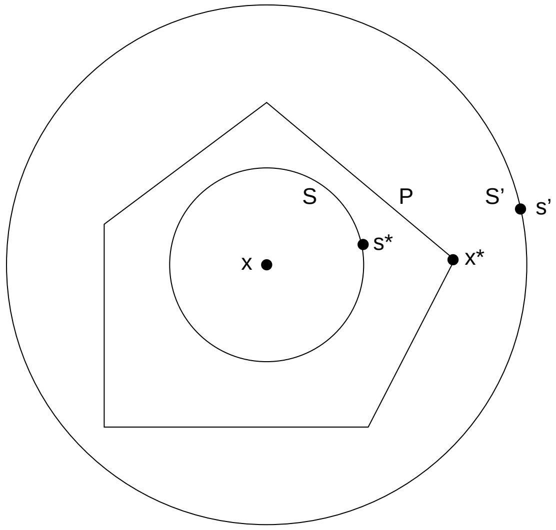
To minimize $3x + 4y$ over the disc with center $(0, 0)$ and radius 2
start at the center & move 2 units along $(-3, -4)$ to $(-6/5, -8/5)$

in general to minimize $\mathbf{cx}$ over a ball of radius r
start at the center & move r units along the vector $-\mathbf{c}$

this works because $\mathbf{cx} = 0$ is the hyperplane of all vectors $\mathbf{x}$ orthogonal to $\mathbf{c}$
so at the point we reach, the hyperplane $\mathbf{cx} = (\text{constant})$ is tangent to the ball

Optimizing Over "Round" Regions

if the feasible region is "round" like a ball, the above strategy should get us close to a minimum



to make this precise suppose we're currently at point $\mathbf{x}$ in the feasible region P
let S (S') be balls contained in (containing) P with center $\mathbf{x}$
let the radius of S' be ρ times that of S, $\rho \geq 1$
let $\mathbf{x}^*$ ($\mathbf{s}^*, \mathbf{s}'$) have minimum cost in P (S, S') respectively

Lemma 1. $\mathbf{cs}^* - \mathbf{cx}^* \leq (1 - 1/\rho)(\mathbf{cx} - \mathbf{cx}^*)$.

Proof. $\mathbf{s}' = \mathbf{x} + \rho(\mathbf{s}^* - \mathbf{x})$. hence

$$\begin{aligned} \mathbf{cx}^* \geq \mathbf{cs}' &= \mathbf{cx} + \mathbf{c}\rho(\mathbf{s}^* - \mathbf{x}) \\ (\rho - 1)\mathbf{cx} + \mathbf{cx}^* &\geq \rho\mathbf{cs}^* \\ (\rho - 1)(\mathbf{cx} - \mathbf{cx}^*) &\geq \rho(\mathbf{cs}^* - \mathbf{cx}^*) \end{aligned}$$

dividing by ρ gives the desired inequality $\square$

Lemma 1 generalizes to allow S to be any closed subset of P:

for $\mathbf{x}$ a point in S, define S' as the scaled up version of S,

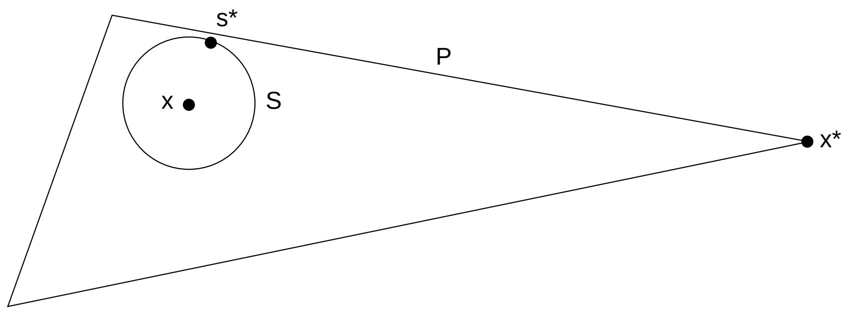
$S' = \{\mathbf{x} + \rho(\mathbf{y} - \mathbf{x}) : \mathbf{y} \in S\}$

assuming $S \subseteq P \subseteq S'$, the same proof works

if ρ is small we get a big improvement by going from $\mathbf{x}$ to $\mathbf{s}^*$

but ρ is big if we're near the boundary of P

we'd be in trouble if we were near the boundary but far from the optimum vertex

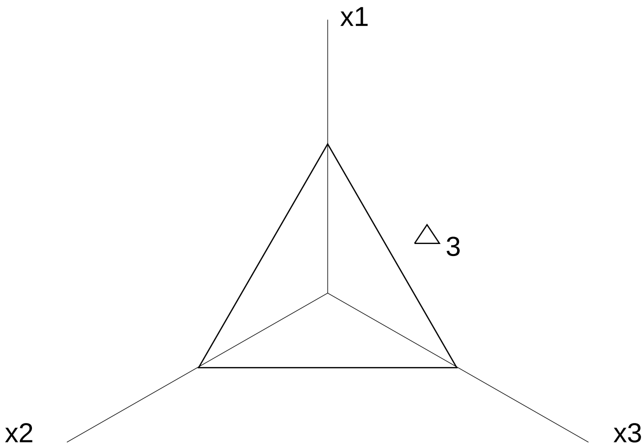


Moving to $\mathbf{s}^*$ makes little progress.

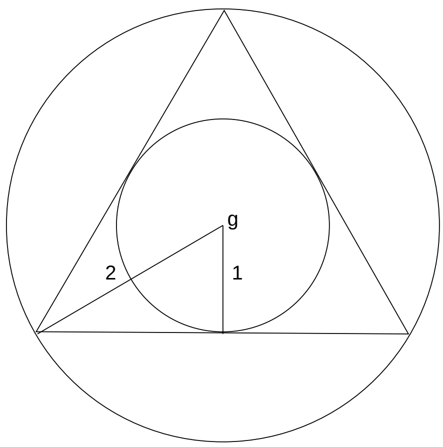
we'll keep ρ relatively small by transforming the problem

we work with simplices because they have small ρ
let $\mathbf{1}$ be the vector of n 1's, $(1,\ldots,1)$

the *standard simplex* Δ_n consists of all $\mathbf{x} \in \mathbf{R}^n$ with $\mathbf{1}^T\mathbf{x} = 1, \mathbf{x} \geq \mathbf{0}$
its *center (of gravity)* is $\mathbf{g} = \mathbf{1}/n$



Lemma 4 of Handout#65 shows that using center $\mathbf{g}$ of Δ_n,
the circumscribed sphere S' has radius $\rho = n-1$ times the radius of the inscribed sphere S
(we'll actually use a slightly different ρ)



$\sin(30^\circ) = \frac{1}{2} \Longrightarrow \rho = 2$ for Δ_3

Karmarkar Standard Form

we always work with simplices, and Karmarkar Standard Form is defined in terms of them

consider the LP

$$\begin{array}{lll} \text{minimize } z = & \mathbf{cx} & \\ \text{subject to} & \mathbf{Ax} & = \mathbf{0} \\ & \mathbf{1}^T\mathbf{x} & = 1 \\ & \mathbf{x} & \ge \mathbf{0} \end{array}$$

$\mathbf{A}$ is an $m \times n$ matrix, all vectors are in $\mathbf{R}^n$

further assume the coefficients in $\mathbf{A}$ & $\mathbf{c}$ are integers

assume that $\mathbf{g}$ is feasible and $z^* \ge 0$

the problem is to find a point with objective value 0 or show that none exists

also assume that the $m+1$ equations are linearly independent

i.e., eliminate redundant rows of $\mathbf{A}$

the exercise of Handout#18 shows that any LP can be converted to this standard form to transform our problem, mapping $\mathbf{p}$ to the center $\mathbf{g}$ of the simplex Δ_n,

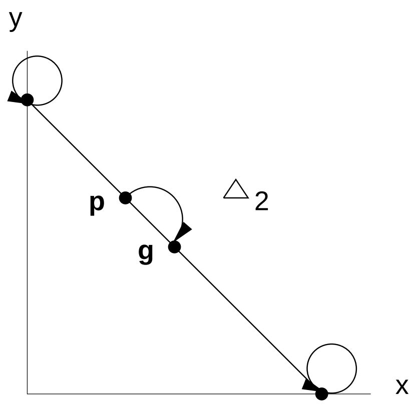
$$(*)\qquad \begin{tikzcd} \mathbf{p} \arrow[r, "T_p"] & \mathbf{g} \arrow[d, "\text{spherical minimization}"] \\ \mathbf{p}' & \mathbf{s} \arrow[l, "T_p^{-1}"] \end{tikzcd}$$

we use the "projective transformation" $\mathbf{y} = T_p(\mathbf{x})$ where

$$y_j = \frac{x_j/p_j}{\sum_{k=1}^n x_k/p_k}$$

here we assume $\mathbf{p} > \mathbf{0}$, $\mathbf{x} \geq \mathbf{0}$ & $\mathbf{x} \neq \mathbf{0}$, so T_p is well-defined

Example.



For Δ_2, $\mathbf{p} = (1/3, 2/3)$, T_p maps (x, y) to $\dfrac{(3x, \frac{3}{2}y)}{3x + \frac{3}{2}y}$.

Since $y = 1 - x$, the point of Δ_2 at x goes to $2 - \dfrac{2}{x+1}$.

Properties of T_p

any vector $\mathbf{y} = T_p(\mathbf{x})$ belongs to Δ_n
since $\mathbf{x} \geq \mathbf{0}$ implies $\mathbf{y} \geq \mathbf{0}$
and the defining formula shows $\mathbf{1}^T\mathbf{y} = 1$

$T_p(\mathbf{p}) = \mathbf{g}$ since all its coordinates are equal

T_p restricted to Δ_n has an inverse since we can recover $\mathbf{x}$:

$$x_j = \frac{y_j p_j}{\sum_{k=1}^n y_k p_k}$$

(the definition of T_p implies $x_j = \gamma y_j p_j$
where the constant γ is chosen to make $\mathbf{1}^T\mathbf{x} = 1$)

Vector notation:

define $\mathbf{D}$ to be the $n \times n$ diagonal matrix with $\mathbf{p}$ along the diagonal
i.e., $D_{jj} = p_j$
the above formulas become

$$\mathbf{x} = \frac{\mathbf{D}\mathbf{y}}{\mathbf{1}^T\mathbf{D}\mathbf{y}} \quad \text{and} \quad \mathbf{y} = \frac{\mathbf{D}^{-1}\mathbf{x}}{\mathbf{1}^T\mathbf{D}^{-1}\mathbf{x}}$$

Exercise. Check T_p is a bijection between Δ_n and Δ_n, i.e., it's onto.

let P be the feasible region of the given LP, i.e., $\mathbf{A}\mathbf{x} = \mathbf{0}$, $\mathbf{1}^T\mathbf{x} = 1$, $\mathbf{x} \geq \mathbf{0}$
let P' be the set of vectors satisfying $\mathbf{A}\mathbf{D}\mathbf{y} = \mathbf{0}$, $\mathbf{1}^T\mathbf{y} = 1$, $\mathbf{y} \geq \mathbf{0}$
let $\mathbf{c}' = \mathbf{c}\mathbf{D}$

Lemma 1. *T_p is a bijection between P and P', with $T_p(\mathbf{p}) = \mathbf{g}$.*
Furthermore $\mathbf{c}\mathbf{x} = 0 \iff \mathbf{c}'T_p(\mathbf{x}) = 0$.

Proof. it remains only to observe that for $\mathbf{y} = T_p(\mathbf{x})$,
$\mathbf{A}\mathbf{x} = \mathbf{0} \iff \mathbf{A}\mathbf{D}\mathbf{y} = \mathbf{0}$, and $\mathbf{c}\mathbf{x} = 0 \iff \mathbf{c}\mathbf{D}\mathbf{y} = 0$. $\square$

our plan $(*)$ is to find $\mathbf{s}$ as minimizing a linear cost function over a ball inscribed in P'

good news: P' is well-rounded
bad news: the cost function in the transformed space is nonlinear,
$\mathbf{c}\mathbf{x} = \mathbf{c}\mathbf{D}\mathbf{y}/(\mathbf{1}^T\mathbf{D}\mathbf{y}) = \mathbf{c}'\mathbf{y}/(\mathbf{1}^T\mathbf{D}\mathbf{y})$

solution: exercising some care, we can ignore the denominator
and minimize $\mathbf{c}'\mathbf{y}$, the *pseudocost*, in transformed space!

Caution. because of the nonlinearity,
the sequence of points $\mathbf{p}$ generated by Karmarkar's algorithm need not have cost $\mathbf{c}\mathbf{p}$ decreasing –
a point $\mathbf{p}'$ may have larger cost than the previous point $\mathbf{p}$

Logarithmic Potential Function

we analyze the decrease in cost $\mathbf{cp}$ using a potential function:

for a vector $\mathbf{x}$, let $\Pi\mathbf{x} = x_1 x_2 \ldots x_n$
the *potential* at $\mathbf{x}$ is

$$f(\mathbf{x}) = \ln\left(\frac{(\mathbf{cx})^n}{\Pi\mathbf{x}}\right)$$

assume $\mathbf{cx} > 0$ & $\mathbf{x} > \mathbf{0}$, as will be the case in the algorithm, so $f(\mathbf{x})$ is well-defined

Remark. f is sometimes called a "logarithmic barrier function" – it keeps us away from the boundary

since $\Pi\mathbf{x} < 1$ in the simplex Δ_n, $f(\mathbf{x}) > \ln(\mathbf{cx})^n$
so pushing $f(\mathbf{x})$ to $-\infty$ pushes $\mathbf{cx}$ to 0

define a corresponding potential in the transformed space, $f_p(\mathbf{y}) = \ln\big((\mathbf{c}'\mathbf{y})^n/\Pi\mathbf{y}\big)$

Lemma 2. *If* $\mathbf{y} = T_p(\mathbf{x})$ *then* $f_p(\mathbf{y}) = f(\mathbf{x}) + \ln(\Pi\mathbf{p})$.

Proof. $f_p(\mathbf{y})$ is the natural log of $(\mathbf{c}'\mathbf{y})^n/\Pi\mathbf{y}$

the numerator of this fraction is $\left(\mathbf{cD}\dfrac{\mathbf{D}^{-1}\mathbf{x}}{\mathbf{1}^T\mathbf{D}^{-1}\mathbf{x}}\right)^n = \left(\dfrac{\mathbf{cx}}{\mathbf{1}^T\mathbf{D}^{-1}\mathbf{x}}\right)^n$

the denominator is $\dfrac{\Pi_{i=1}^n(x_i/p_i)}{(\mathbf{1}^T\mathbf{D}^{-1}\mathbf{x})^n}$

so the fraction equals $\dfrac{(\mathbf{cx})^n}{\Pi_{i=1}^n(x_i/p_i)} = \dfrac{(\mathbf{cx})^n}{\Pi\mathbf{x}}\Pi\mathbf{p}$

taking its natural log gives the lemma $\square$

in the scheme $(*)$ we will choose $\mathbf{s}$ so $f_p(\mathbf{s}) \le f_p(\mathbf{g}) - \delta$, for some positive constant δ
the Lemma shows we get $f(\mathbf{p}') \le f(\mathbf{p}) - \delta$

thus each step of the algorithm decreases $f(\mathbf{p})$ by δ
and we push the potential to $-\infty$ as planned

recall the parameter $L = mn + n\lceil \log n\rceil + \sum\{\lceil \log |r|\rceil : r \text{ a nonzero entry in } \mathbf{A} \text{ or } \mathbf{c}\}$ (see Handout #25)

Karmarkar's Algorithm
Initialization
Set $\mathbf{p} = \mathbf{1}/n$. If $\mathbf{cp} = 0$ then return $\mathbf{p}$.
Let $\delta > 0$ be a constant determined below (Handout#70, Lemma 4). Set $N = \lceil 2nL/\delta \rceil$.

Main Loop
Repeat the Advance Step N times (unless it returns).
Then go to the Rounding Step.

Advance Step
Advance from $\mathbf{p}$ to the next point $\mathbf{p}'$, using an implementation of $(*)$.
If $\mathbf{cp}' = 0$ then return $\mathbf{p}'$.
Set $\mathbf{p} = \mathbf{p}'$.

Rounding Step
Move from $\mathbf{p}$ to a vertex $\mathbf{v}$ of no greater cost. (Use the exercise of Handout#23.)
If $\mathbf{cv} = 0$ then return $\mathbf{v}$, else return "$z^* > 0$".

a *valid implementation* of $(*)$ has these properties:
assume $z^* = 0$, $\mathbf{p} \in P$, $\mathbf{p} > \mathbf{0}$ & $\mathbf{cp} > 0$
then $\mathbf{p}' \in P$, $\mathbf{p}' > \mathbf{0}$, and either $\mathbf{cp}' = 0$ or $f(\mathbf{p}') \le f(\mathbf{p}) - \delta$

Lemma 1. *A valid implementation of* $(*)$ *ensures the algorithm is correct.*

Proof. we can assume the Main Loop repeats N times

we start at potential value $f(\mathbf{g}) = \ln\left((\mathbf{c1}/n)^n/(1/n)^n\right) = n\ln\left(\sum_{i=1}^n c_i\right) \le nL$
the last inequality follows since if C is the largest cost coefficient,
$\ln\left(\sum_{i=1}^n c_i\right) \le \ln(nC) \le \ln n + \ln C \le L$

each repetition decreases the potential by $\ge \delta$
so the Main Loop ends with a point $\mathbf{p}$ of potential $\le nL - N\delta \le -nL$
thus $\ln(\mathbf{cp})^n < f(\mathbf{p}) < -nL$, $\quad \mathbf{cp} < e^{-L} < 2^{-L}$

so the Rounding Step finds a vertex of cost $\gamma < 2^{-L}$
γ is a rational number with denominator $< 2^L$ (by the exercise of Handout#25)
so $\gamma > 0 \Longrightarrow \gamma > 1/2^L$
thus $\gamma = 0$ $\quad\square$

Idea for Implementing $(*)$

as in Handout#68, we need to go from $\mathbf{g}$ to a point $\mathbf{s}$
where $f_p(\mathbf{s}) \le f_p(\mathbf{g}) - \delta$

since $f_p(\mathbf{s})$ is the log of $(\mathbf{c}'\mathbf{s})^n/\Pi\mathbf{s}$
we could define $\mathbf{s}$ to minimize $\mathbf{c}'\mathbf{s}$ over the inscribed ball S

but to prevent the denominator from decreasing too much we use a slightly smaller ball:
S has radius $r = 1/\sqrt{n(n-1)} > 1/n$ (Handout #65, Lemma 4)
minimize over the ball of radius α/n, for some value $\alpha \le 1$
actually Lemma 4 of Handout#70 shows that α must be $< 1/2$

Implementation of $(*)$

Let $\mathbf{B}$ be the $(m+1)\times n$ matrix $\begin{bmatrix}\mathbf{AD}\\ \mathbf{1}^T\end{bmatrix}$

Let $\mathbf{c}'$ be the pseudocost vector $\mathbf{cD}$

Project $\mathbf{c}'$ onto the nullspace of $\mathbf{B}$ to get $\mathbf{c}_P$:
$\mathbf{c}_P = \mathbf{c}' - \mathbf{B}^T(\mathbf{B}\mathbf{B}^T)^{-1}\mathbf{B}\mathbf{c}'$

If $\mathbf{c}_P = \mathbf{0}$ then return "$z^* > 0$". Otherwise move α/n units in the direction $-\mathbf{c}_P$:
$\mathbf{s} = \mathbf{g} - (\alpha/n)\mathbf{c}_P/\|\mathbf{c}_P\|$

Return to the original space:
$\mathbf{p}' = \mathbf{Ds}/(\mathbf{1}^T\mathbf{Ds})$

Exercise. What is right, and what is wrong, with Professor Dull's objection to our implementation:

> "I doubt this implementation will work. The plan was to minimize over an inscribed ball. The implementation minimizes over a ball S in transformed space. But in real space it's minimizing over $T_p^{-1}(S)$, which is not a ball."

Remark.
the projection step can be implemented more carefully:
(i) as a rank 1 modification from the previous projection step
this achieves $O(n^{2.5})$ arithmetic operations, rather than $O(n^3)$
(ii) to take advantage of sparsity of $\mathbf{A}$ (& $\mathbf{B}$)

we prove the implementation of $(*)$ is valid in 5 lemmas

Lemma 1. *The formula for projecting* $\mathbf{c}'$ *onto the nullspace of* $\mathbf{B}$ *is correct.*

Proof.
the rows of $\mathbf{B}$ are linearly independent
since standard form assumes $\mathbf{A}$ and $\mathbf{1}^T$ are linearly independent
so the lemma follows from Lemma 1 of Handout#65 $\square$

Lemma 2. $\mathbf{s}$ *minimizes the cost function* $\mathbf{c}'\mathbf{x}$ *over* $B(\mathbf{g}, \alpha/n) \cap P'$.

Proof.
the lemma follows from Lemma 2 of Handout#65
if we show that $B(\mathbf{g}, \alpha/n) \cap P'$ is the intersection of a ball and an affine space

let F be the affine space of points satisfying $\mathbf{ADx} = \mathbf{0}$, $\mathbf{1}^T\mathbf{x} = 1$

Claim: $B(\mathbf{g}, \alpha/n) \cap P' = B(\mathbf{g}, \alpha/n) \cap F$

comparing the definitions of F & P' (Handout#68), it suffices to show
any coordinate of a point in $B(\mathbf{g}, \alpha/n)$ is nonnegative
this follows since any coordinate is $\geq g_j - \alpha/n = 1/n - \alpha/n \geq 0$ $\square$

Lemma 3. $z^* = 0 \Longrightarrow \mathbf{c}_P \neq \mathbf{0}$.

Proof.
suppose $\mathbf{c}_P = \mathbf{0}$
then the formula for $\mathbf{c}_P$ shows $\mathbf{c}'$ is in the rowspace of B
$\therefore$ $\mathbf{c}'$ is orthogonal to every vector in the nullspace of $\mathbf{B}$

take any $\mathbf{q} \in P$
thus $T_p(\mathbf{q}) \in P'$, and $T_p(\mathbf{q}) - T_p(\mathbf{p})$ is in the nullspace of $\mathbf{B}$

so $\mathbf{c}'(T_p(\mathbf{q}) - T_p(\mathbf{p})) = 0$
recalling from Handout#68, Lemma 1 how cost $\mathbf{c}$ transforms to $\mathbf{c}'$,
$\mathbf{cp} > 0 \implies \mathbf{c}'T_p(\mathbf{p}) > 0$
thus $\mathbf{c}'T_p(\mathbf{q}) > 0$, and so $\mathbf{cq} > 0$
equivalently, $z^* > 0$ $\square$

Lemma 4. $z^* = 0 \Longrightarrow \mathbf{c}'\mathbf{s}/(\mathbf{c}'\mathbf{g}) < 1 - \alpha/n$.

Proof. we apply Lemma 1 of Handout#66:

the inscribed set S is $B(\mathbf{g}, \alpha/n) \cap F$
Lemma 2 above shows we optimize over this set

the circumscribed set S' is $B(\mathbf{g}, R) \cap F$
clearly this set contains the feasible region P'

S' is S scaled up by the factor $\rho = R/(\alpha/n) < n/\alpha$

since $R = \sqrt{(n-1)/n} < 1$ (Handout #65, Lemma 4)

now assuming $z^* = 0$ the lemma gives $\mathbf{c's^*} \le (1 - 1/\rho)\mathbf{c'g} \le (1 - \alpha/n)\mathbf{c'g}$ $\square$

Lemma 5. *Choosing α as an arbitrary real value in $(0, 1/2)$ and $\delta = \alpha - \alpha^2/(1-\alpha)$ gives a valid implementation of $(*)$.*

Proof.
the first 2 requirements for validity are clear:

$\mathbf{p}' \in P$ since $\mathbf{s} \in P'$ (we're using Lemma 4 of Handout#65 & Lemma 1 of Handout#68!)

$\mathbf{p}' > \mathbf{0}$ (since $\alpha < 1$, each coordinate s_j is positive)

for the 3rd requirement, assume $z^* = 0$, $\mathbf{cp}' > 0$
we must show $f(\mathbf{p}') \le f(\mathbf{p}) - \delta$
from Handout#68,p.2 this means $f_p(\mathbf{s}) \le f_p(\mathbf{g}) - \delta$

by definition $f_p(\mathbf{y}) = \ln\left((\mathbf{c'y})^n/\Pi\mathbf{y}\right)$

this gives $f_p(\mathbf{s}) - f_p(\mathbf{g}) = \ln\left(\dfrac{\mathbf{c's}}{\mathbf{c'g}}\right)^n - \ln\left(\Pi(n\mathbf{s})\right)$

the 1st term is $\le -\alpha$:

$$\ln\left(\frac{\mathbf{c's}}{\mathbf{c'g}}\right)^n \underset{\substack{\uparrow \\ \text{Lemma 4}}}{<} \ln(1-\alpha/n)^n = n\ln(1-\alpha/n) \le -\alpha$$

the 2nd term is $\le \alpha^2/(1-\alpha)$:

apply the Lemma 3 of Handout#65 to vector $\mathbf{x} = n\mathbf{s}$

$\mathbf{s} > 0$

$\sum_{j=1}^n s_j = 1$

$\|\mathbf{1} - n\mathbf{s}\| = n\|\mathbf{1}/n - \mathbf{s}\| = n\alpha/n = \alpha < 1$

conclude $\ln\left(\Pi(n\mathbf{s})\right) \ge \dfrac{\alpha^2}{\alpha - 1}$

combining the 2 estimates, $f_p(\mathbf{s}) - f_p(\mathbf{g}) \le -\alpha + \alpha^2/(1-\alpha)$

the lemma chooses δ as the negative of the r.h.s., giving the desired inequality
furthermore choosing $\alpha < 1/2$ makes $\delta > 0$ $\square$

Theorem. *Karmarkar's algorithm solves an LP in polynomial time, assuming all arithmetic operations are carried out exactly.*

Proof.
the Main Loop repeats $O(nL)$ times
each repetition performs all matrix calculations in $O(n^3)$ arithmetic operations

including taking a square root to calculate $\|c_P\|$ in $(*)$

so we execute $O(n^4L)$ arithmetic operations
(Vaidya (*STOC '90*) reduces this to $O(n^3L)$) $\square$

it can be proved that maintaining $O(L)$ bits of precision is sufficient
thus completing the proof of a polynomial time bound

we solve the exercises of Handout#65 for Karmarkar's algorithm

Exercise 1:

Lemma 3. *Let $\mathbf{x} \in \mathbf{R}^n$ be a vector with $\mathbf{x} > \mathbf{0}$ and $\sum_{j=1}^n x_j = n$. Set $\alpha = \|\mathbf{1} - \mathbf{x}\|$ & assume $\alpha < 1$. Then*

$$\ln\left(\prod_{j=1}^n x_j\right) \geq \frac{\alpha^2}{\alpha - 1}.$$

Proof.
since the geometric mean is at most the arithmetic mean,

$\prod_{j=1}^n 1/x_j \leq [\sum_{j=1}^n (1/x_j)/n]^n$

let $\mathbf{y} = \mathbf{1} - \mathbf{x}$
so the r.h.s. becomes $[\sum_{j=1}^n (1/(1-y_j))/n]^n$

we will upperbound the sum in this expression, using these properties of y_j:

$\sum_j y_j = 0$
$\|\mathbf{y}\| = \alpha$
for each j, $|y_j| < 1$ (since $|y_j| \leq \|\mathbf{y}\| = \alpha < 1$)

so the sum is

$$\begin{aligned}
\sum_{j=1}^n (1/(1-y_j)) &\leq \sum_{j=1}^n (1 + y_j + y_j^2 + y_j^3 + y_j^4 + \ldots) \\
&= \sum_{j=1}^n (1 + y_j) + \sum_{j=1}^n (y_j^2 + y_j^3 + y_j^4 + \ldots) \\
&= n + 0 + \sum_{j=1}^n y_j^2 (1 + y_j + y_j^2 + \ldots) \\
&\leq n + \sum_{j=1}^n y_j^2 (1 + \alpha + \alpha^2 + \ldots) \\
&= n + \sum_{j=1}^n y_j^2/(1-\alpha) \\
&= n + \|\mathbf{y}\|^2/(1-\alpha) \\
&= n + \alpha^2/(1-\alpha)
\end{aligned}$$

we have shown $\prod_{j=1}^n 1/x_j \leq [1 + \alpha^2/n(1-\alpha)]^n$

taking logs,

$\ln \prod_{j=1}^n 1/x_j \leq n \ln [1 + \alpha^2/n(1-\alpha)] \leq n\alpha^2/n(1-\alpha) = \alpha^2/(1-\alpha)$ □

Exercise 2:

Lemma 4. *Let H be the hyperplane $\sum_{i=1}^{n} x_i = 1$. Let Δ be the subset of H where all coordinates x_i are nonnegative. Let $\mathbf{g} = (1/n, \ldots, 1/n)$.*

(i) Any point in Δ is at distance at most $R = \sqrt{(n-1)/n}$ from $\mathbf{g}$.

(ii) Any point of H within distance $r = 1/\sqrt{n(n-1)}$ of $\mathbf{g}$ is in Δ.

Proof.
(i) take any point in $\mathbf{x} \in \Delta_n$
we show its distance from $\mathbf{g}$ is $\leq R$ by starting at $\mathbf{x}$,
moving away from $\mathbf{g}$, and eventually reaching a corner point like $(1, 0, \ldots, 0)$,
which we've seen is at distance R

wlog let x_1 be the maximum coordinate x_j
choose any positive x_j, $j > 1$
increase x_1 by x_j and decrease x_j to 0
we stay on H,
this increases the distance from $\mathbf{g}$, since $(x - 1/n)^2$ is concave up
repeat this until the corner point $(1, 0, \ldots, 0)$ is reached

(ii) it suffices to show any point $\mathbf{x} \in H$ with $x_n < 0$ is at distance $> r$ from $\mathbf{g}$

a point of H with $x_n < 0$ has $\sum_{j=1}^{n-1} x_j > 1$
to minimize $\sum_{j=1}^{n-1} (x_j - 1/n)^2$, set all coordinates equal (by Jensen)
so the minimum sum is $> (n-1)(1/(n-1) - 1/n)^2$

this implies the distance to $\mathbf{g}$ is
$\sum_{j=1}^{n} (x_j - 1/n)^2 > (n-1)(1/(n-1) - 1/n)^2 + 1/n^2 = r^2$ $\square$

source: *Primal-dual Interior-point Methods* by S.J. Wright, SIAM, Philadelphia PA, 1997.

the break-through papers:
L.G. Khachiyan, "A polynomial algorithm in linear programming," *Soviet Math. Dolkady*, 1979
N. Karmarkar, "A new polynomial-time algorithm for linear programming," *Combinatorica*, 1984

after Karmarkar other interior-point methods were discovered,
making both theoretic and practical improvements

Primal-dual Interior-point Methods

consider an LP

$$\begin{aligned} &\text{minimize} && \mathbf{cx} \\ &\text{subject to} && \mathbf{Ax} = \mathbf{b} \\ &&& \mathbf{x} \geq \mathbf{0} \end{aligned}$$

& its dual,

$$\begin{aligned} &\text{maximize} && \mathbf{yb} \\ &\text{subject to} && \mathbf{yA} + \mathbf{s} = \mathbf{c} \\ &&& \mathbf{s} \geq \mathbf{0} \end{aligned}$$

we find optimum primal and dual solutions by solving the system $(*)$ of Handout#19, p.2:

$$(*) \qquad \begin{aligned} \mathbf{Ax} &= \mathbf{b} \\ \mathbf{yA} + \mathbf{s} &= \mathbf{c} \\ \mathbf{x}_j \mathbf{s}_j &= 0 \qquad j = 1, \ldots, n \\ \mathbf{x}, \mathbf{s} &\geq \mathbf{0} \end{aligned}$$

Approach
apply Newton's method to the 3 equations of $(*)$
modified so that "positivity" $(\mathbf{x}, \mathbf{s} > \mathbf{0})$ always holds
this keeps us in the interior and avoids negative values!

the *complementary slackness measure* $\mu = \sum_{j=1}^{n} x_j s_j$

there are 2 approaches for the modification

Potential-reduction methods
each step reduces a logarithmic potential function

the potential function has 2 properties:
(a) it approaches ∞ if $x_j s_j \to 0$ for some j but $\mu \not\to 0$
(b) it approaches $-\infty$ if & only if $(\mathbf{x}, \mathbf{y}, \mathbf{s})$ approaches an optimum point

a very good potential function: $\rho \ln \mu \; - \; \sum_{j=1}^{n} \ln (\mathbf{x}_j \mathbf{s}_j)$
where ρ is a parameter $> n$

note the similarity to Karmarkar's potential function!

Path-following methods
the *central path* of the feasible region is a path $(\mathbf{x}_t, \mathbf{y}_t, \mathbf{s}_t)$ (t is a parameter > 0)
satisfying $(*)$ with the 3rd constraint replaced by

$$\mathbf{x}_j \mathbf{s}_j = t, \quad j = 1, \ldots, n$$

clearly this implies positivity
as t approaches 0 we approach the desired solution

we can take bigger Newton steps along the central path

predictor-corrector methods alternate between 2 types of steps:
(a) a predictor step: a pure Newton step, reducing μ
(b) a corrector step: moves back closer to the central path

Mehrotra's predictor-corrector algorithm is the basis of most current interior point codes
e.g., CPLEX

Infeasible-interior-point methods can start without a feasible interior point

extensions of these methods solve semidefinite programming (Handouts#41, 44)
& *(convex) quadratic programming*,

$$\begin{array}{ll} \text{minimize} & \mathbf{c}^T\mathbf{x} + \mathbf{x}^T\mathbf{Q}\mathbf{x} \\ \text{subject to} & \mathbf{A}\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{array}$$

where $\mathbf{Q}$ is symmetric positive semidefinite (Handouts#42, 43)

LCP: we are given an $n \times n$ matrix $\mathbf{A}$ & a length n column vector $\mathbf{b}$
we wish to find length n column vectors $\mathbf{x}$, $\mathbf{y}$ satisfying

$$\begin{aligned} \mathbf{y} - \mathbf{A}\mathbf{x} &= \mathbf{b} \\ \mathbf{y}^T\mathbf{x} &= 0 \\ \mathbf{x}, \mathbf{y} &\geq \mathbf{0} \end{aligned}$$

equivalently for each $i = 1, \ldots, n$, discard 1 of y_i, x_i
then find a nonnegative solution to the reduced linear system

by way of motivation we show LCP generalizes LP.

Proof. consider a primal-dual pair
max $\mathbf{cx}$ s.t. $\mathbf{Ax} \leq \mathbf{b},\ \mathbf{x} \geq \mathbf{0}$ min $\mathbf{yb}$ s.t. $\mathbf{yA} \geq \mathbf{c},\ \mathbf{y} \geq \mathbf{0}$
introduce primal slacks $\mathbf{s}$ and dual slacks $\mathbf{t}$
$\therefore$ $\mathbf{x}$ & $\mathbf{y}$ are optimum $\iff \mathbf{s}, \mathbf{t} \geq \mathbf{0}$ & $\mathbf{ys} = \mathbf{tx} = 0$
we can rewrite the optimality condition as this LCP:

$$\begin{bmatrix} \mathbf{s} \\ \mathbf{t}^T \end{bmatrix} - \begin{bmatrix} \mathbf{0} & -\mathbf{A} \\ \mathbf{A}^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{y}^T \\ \mathbf{x} \end{bmatrix} = \begin{bmatrix} \mathbf{b} \\ -\mathbf{c}^T \end{bmatrix}$$

$$\begin{bmatrix} \mathbf{s}^T & \mathbf{t} \end{bmatrix} \begin{bmatrix} \mathbf{y}^T \\ \mathbf{x} \end{bmatrix} = 0$$

$$\mathbf{s}, \mathbf{t}, \mathbf{x}, \mathbf{y} \geq \mathbf{0} \qquad \square$$

Exercise. Explain why LCP is a special case of QP (Handout#42)

algorithms such as complementary pivot (simplex) algorithm and interior point solve LCP

Application to Game Theory: Bimatrix Games

also called *nonzero sum 2-person games*

we have two $m \times n$ payoff matrices $\mathbf{A}$, $\mathbf{B}$
if `ROW` player chooses i & `COLUMN` chooses j, `ROW` loses a_{ij} & `COLUMN` loses b_{ij}

`ROW` plays according to a stochastic column vector $\mathbf{x}$ (length m)
`COLUMN` plays according to a stochastic column vector $\mathbf{y}$ (length n)
so `ROW` has expected loss $\mathbf{x}^T\mathbf{A}\mathbf{y}$, `COLUMN` has expected loss $\mathbf{x}^T\mathbf{B}\mathbf{y}$

Example. In the game of *Chicken*, whoever chickens out first loses

neither player chickens out →	$2,2$	$-1,1$	
	$1,-1$	$0,0$	← both players chicken out

in a bimatrix game $\mathbf{x}^*, \mathbf{y}^*$ form a Nash equilibrium point (recall Handout#22) if

$$\mathbf{x}^{*T}\mathbf{A}\mathbf{y}^* \le \mathbf{x}^T\mathbf{A}\mathbf{y}^* \text{ for all stochastic vectors } \mathbf{x} \quad (\texttt{ROW} \text{ can't improve})$$
$$\mathbf{x}^{*T}\mathbf{B}\mathbf{y}^* \le \mathbf{x}^{*T}\mathbf{B}\mathbf{y} \text{ for all stochastic vectors } \mathbf{y} \quad (\texttt{COLUMN} \text{ can't improve})$$

Example. Chicken has 2 pure Nash points: `ROW` always chickens out & `COLUMN` never does, and vice versa Also, both players choose randomly with probability 1/2.
$(2(1/2) - 1(1/2) = 1/2,\ 1(1/2) + 0(1/2) = 1/2)$

Fact. Any bimatrix game has a (stochastic) Nash point.

Theorem. *The Nash equilibria correspond to the solutions of an LCP.*

Proof.

1. can assume all entries of $\mathbf{A}$ & $\mathbf{B}$ are > 0

Proof. for any $\mathbf{A}'$, distributivity shows $\mathbf{x}^T(\mathbf{A}+\mathbf{A}')\mathbf{y} = \mathbf{x}^T\mathbf{A}\mathbf{y} + \mathbf{x}^T\mathbf{A}'\mathbf{y}$
any stochastic $\mathbf{x}$, $\mathbf{y}$ have $\mathbf{x}^T\mathbf{1}\mathbf{y} = 1$
for $\mathbf{1}$ an $m \times n$ matrix of 1's
so we can increase $\mathbf{A}$ by a large multiple of $\mathbf{1}$,
without changing the Nash equilibria, to make every coefficient positive $\diamondsuit$

2. let $\mathbf{x}^*, \mathbf{y}^*$ be a Nash equilibrium
rewrite the Nash conditions: (we'll do it for `ROW`'s condition & $\mathbf{A}$; `COLUMN` & $\mathbf{B}$ is symmetric)
write $\ell = \mathbf{x}^{*T}\mathbf{A}\mathbf{y}^*$

(a) taking $x_i = 1$ & all other $x_j = 0$ shows each component $(\mathbf{A}\mathbf{y}^*)_i$ of $\mathbf{A}\mathbf{y}^*$ must be $\ge \ell$

(b) since $\mathbf{x}^{*T}\mathbf{A}\mathbf{y}^* = \sum_i \mathbf{x}_i^*(\mathbf{A}\mathbf{y}^*)_i \ge \sum_i \mathbf{x}_i^*\ell = \ell$
we must have for all i, either $\mathbf{x}_i^* = 0$ or $(\mathbf{A}\mathbf{y}^*)_i = \ell$

it's easy to see that conversely, (a) & (b) guarantee the Nash condition for `ROW`

assumption #1 with $\mathbf{x}^*, \mathbf{y}^*$ stochastic implies $\ell > 0$
so we can define vectors

$$\overline{\mathbf{x}} = \frac{\mathbf{x}^*}{\mathbf{x}^{*T}\mathbf{B}\mathbf{y}^*}, \quad \overline{\mathbf{y}} = \frac{\mathbf{y}^*}{\mathbf{x}^{*T}\mathbf{A}\mathbf{y}^*}$$

(a) becomes $\mathbf{A}\overline{\mathbf{y}} \ge \mathbf{1}$ ($\mathbf{1}$ is a column vector of 1's)
letting $\mathbf{u}$ be the vector of slacks in this inequality,
(b) becomes $\mathbf{u}^T\overline{\mathbf{x}} = 0$

so we've shown $\overline{\mathbf{x}}, \overline{\mathbf{y}}$, $\mathbf{u}$ & $\mathbf{v}$ (the slacks for $\overline{\mathbf{x}}$) satisfy this LCP:

$$\begin{aligned}
\begin{bmatrix}\mathbf{u}\\ \mathbf{v}\end{bmatrix} - \begin{bmatrix}\mathbf{0} & \mathbf{A}\\ \mathbf{B}^T & \mathbf{0}\end{bmatrix}\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix} &= -\mathbf{1}\\
\begin{bmatrix}\mathbf{u}\\ \mathbf{v}\end{bmatrix}^T\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix} &= 0\\
\mathbf{x}, \mathbf{u}, \mathbf{y}, \mathbf{v} &\ge \mathbf{0}
\end{aligned}$$

3. conversely let $\mathbf{x}$, $\mathbf{y}$ be any solution to the LCP
make them stochastic vectors:

$$\mathbf{x}^* = \frac{\mathbf{x}}{\mathbf{1}^T\mathbf{x}}, \quad \mathbf{y}^* = \frac{\mathbf{y}}{\mathbf{1}^T\mathbf{y}}$$

these vectors form a Nash equilibrium point, by an argument similar to #2
   e.g., we know $\mathbf{A}\mathbf{y} \ge \mathbf{1}$
   also $\mathbf{x}^T\mathbf{A}\mathbf{y} = \mathbf{1}^T\mathbf{x}$ by complementarity $\mathbf{u}^T\mathbf{x} = \mathbf{0}$
   thus $\mathbf{x}^{*T}\mathbf{A}\mathbf{y} = 1, \quad \mathbf{A}\mathbf{y} \ge (\mathbf{x}^{*T}\mathbf{A}\mathbf{y})\mathbf{1}, \quad \mathbf{A}\mathbf{y}^* \ge (\mathbf{x}^{*T}\mathbf{A}\mathbf{y}^*)\mathbf{1}$
   the last inequality implies `ROW`'s Nash condition $\square$

the dual is defined using the constraints of the KKT conditions:

Primal
minimize $\frac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{c}\mathbf{x}$
subject to $\mathbf{A}\mathbf{x} \geq \mathbf{b}$
$\mathbf{x} \geq \mathbf{0}$

there are 2 row vectors of dual variables –
$\mathbf{y}$, an m-vector of duals corresponding to the m linear constraints (i.e., the LP duals)
$\mathbf{z}$, an n-vector of free variables, corresponding to the objective function's $\mathbf{Q}$

Dual
maximize $-\frac{1}{2}\mathbf{z}\mathbf{Q}\mathbf{z}^T + \mathbf{y}\mathbf{b}$
subject to $\mathbf{y}\mathbf{A} + \mathbf{z}\mathbf{Q} \leq \mathbf{c}$
$\mathbf{y} \geq \mathbf{0}$

Theorem. *(Weak Duality) If* $\mathbf{Q}$ *is PSD, any feasible dual (max) solution lower bounds any feasible primal (min) solution.*

Proof.
PSD gives $(\mathbf{z} + \mathbf{x}^T)\mathbf{Q}(\mathbf{z}^T + \mathbf{x}) \geq 0$, i.e.,
$\mathbf{z}\mathbf{Q}\mathbf{z}^T + 2\mathbf{z}\mathbf{Q}\mathbf{x} + \mathbf{x}^T\mathbf{Q}\mathbf{x} \geq 0$

as in LP Weak Duality we have
$\mathbf{y}\mathbf{b} \leq \mathbf{y}\mathbf{A}\mathbf{x}$
& rewriting the PSD inequality gives
$-\frac{1}{2}\mathbf{z}\mathbf{Q}\mathbf{z}^T \leq \mathbf{z}\mathbf{Q}\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x}$

adding these 2 inequalities, the dual objective is on the left
the first 2 terms on the right are upper bounded as in LP Weak Duality:
$\mathbf{y}\mathbf{A}\mathbf{x} + \mathbf{z}\mathbf{Q}\mathbf{x} \leq \mathbf{c}\mathbf{x}$
giving the primal objective on the right, i.e.,
(primal objective) $\leq$ (dual objective) $\square$

Exercise. Prove Strong Duality.

Examples.
for simplicity we'll use 1-dimensional primals

1. Primal: min $x^2/2$ s.t. $x \geq 1$
$\mathbf{Q} = (1)$, PD
the optimum solution is $x = 1$, objective $= 1/2$

Dual: max $-z^2/2 + y$ s.t. $y + z \leq 0, y \geq 0$
optimum solution $y = 1, z = -1$, objective $= -1/2 + 1 = 1/2$

Proof this dual solution is optimum:
$z \leq -y \leq 0 \implies z^2 \geq y^2$, $-z^2/2 \leq -y^2/2$
$\therefore$ objective $\leq -y^2/2 + y = y(-y/2 + 1)$
this quadratic achieves its maximum midway between the 2 roots, i.e., $y = 1$ $\square$

2. we show Weak Duality fails in an example where $\mathbf{Q}$ is not PSD:
use Example 1 except $\mathbf{Q} = (-1)$

Primal: min $-x^2/2$ s.t. $x \geq 1$
the problem is unbounded

Dual: max $z^2/2 + y$ s.t. $y - z \leq 0, y \geq 0$
taking $y = z$ shows the problem is unbounded

so the primal does not upper bound the dual

1. If $\mathbf{Q}$ is not PSD a point satisfying KKT need not be optimum. In fact every vertex can be a local min! No good QP algorithms are known for this general case.

2. this isn't surprising: QP is NP-hard
   integer QP is undecidable!

we give an example where $\mathcal{Q}$ with $\mathbf{Q}$ PD has a unique optimum
but flipping $\mathbf{Q}$'s sign makes every vertex a local min!

$\mathbf{Q}$ PD:
$\mathcal{Q}$: min $\sum_{j=1}^{n} x_j(1+x_j) + \sum_{j=1}^{n} \delta_j x_j$ s.t. $0 \le x_j \le 1$, $j = 1, \ldots, n$
assume the δ's are "small", specifically for each j, $|\delta_j| < 1$

going to standard form, $\mathbf{A} = -\mathbf{I}$, $\mathbf{b} = (-1, \ldots, -1)^T$, $\mathbf{Q} = 2\mathbf{I}$

the dual KKT constraint $\mathbf{A}^T\mathbf{y} - \mathbf{Q}\mathbf{x} \le \mathbf{c}^T$ is
$-y_j - 2x_j \le 1 + \delta_j$, $j = 1, \ldots, n$
the KKT CS constraints are
$x_j > 0 \implies -y_j - 2x_j = 1 + \delta_j$, i.e., $2x_j = -1 - \delta_j - y_j$
$y_j > 0 \implies x_j = 1$

the first CS constraint implies we must have $x_j = 0$ for all j
(since our assumption on δ_j implies $-1 - \delta_j - y_j < -y_j \le 0$)
taking $\mathbf{x} = \mathbf{y} = \mathbf{0}$ satisfies all KKT conditions, so the origin is the unique optimum
(as expected)

$\mathbf{Q}$ ND:
flipping $\mathbf{Q}$'s sign is disastrous – we get an exponential number of solutions to KKT!

$\mathcal{Q}$: flip the sign of the quadratic term in the objective function, $x_j(1+x_j) \to x_j(1-x_j)$

now $\mathbf{Q} = -2\mathbf{I}$ but the other matrices & vectors are unchanged

the dual KKT constraint gets a sign flipped,
$-y_j + 2x_j \le 1 + \delta_j$, $j = 1, \ldots, n$
& the KKT CS constraints change only in that sign:
$x_j > 0 \implies -y_j + 2x_j = 1 + \delta_j$, i.e., $2x_j = 1 + \delta_j + y_j$
$y_j > 0 \implies x_j = 1$

the new KKT system has 3^n solutions:
there are 3 possibilities for each j,
$x_j = y_j = 0$
$x_j = 1$, $y_j = 1 - \delta_j$
$x_j = (1 + \delta_j)/2$, $y_j = 0$

it's easy to check all 3 satisfy all KKT conditions
note the 3rd alternative is the only possibility allowed by CS when $0 < x_j < 1$
the feasible region has 2^n vertices, each is a KKT solution,
and it can be shown that each vertex is a local minimum