Optimization Pg 1

I. Introduction
II. Linear Systems
III. Integer Linear Programming

Chapter 4, 4.0 – 4.4

I. Introduction

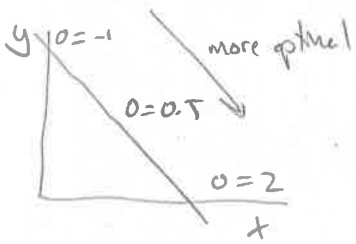
1. Example with two variables:

O = Objective function

$O = 2x - y$

$x + y = 1$

$x, y \geq 0$



Can generalize to three variables

$O = 3x + y + z$

$x + y + z = 1$

$x, y, z \geq 0$

II. Linear systems

$$\min \; C^T x \quad \text{s.t.} \quad Gx \leq h, \; Ax = b$$

1. Max. O

$O = c_1 x_1 + c_2 x_2 + c_3 x_3 \ldots + c_n x_n$

O is a linear combination of control parameters

c_is are real scalars

x_is are constrained

$$a_{11} x_1 + a_{12} x_2 + \ldots \ldots a_{1n} x_n = b_1$$
$$a_{21} x_1 + a_{22} x_2 + \ldots \ldots a_{2n} x_n = b_2$$
$$\vdots$$
$$a_{m1} x_1 + a_{m2} x_2 + \ldots a_{mn} x_n = b_m$$

In matrix notation

$$Ax = b$$

A is an $m \times n$ matrix

x, b are column vectors with n and m elements, with $n \geq m$

System must be underconstrained

$r < n$

Note that:

if a constraint is an inequality:

$a_{i1} x_1 + a_{i2} x_2 \ldots a_{in} x_n \geq b_i$

we can re-write using a surplus variable:

$$a_{i1} x_1 + a_{i2} x_2 \ldots a_{in} x_n + s_i = b_i$$

2. Coins example (ILP)

	Penny	Nickel	Dime	Quarter	Dollar
Copper (Cu)	0.06g	3.8g	2.1g	5.2g	7.2g
Nickel (Ni)		1.2g	0.2g	0.5g	0.2g
Zinc (Zi)	2.4g				
Manganese (Mn)					0.5g
					0.3g

Assume we have:

1000 g of copper

50 g of other materials

what coins should we make?

Optimization Pg 2

To formulate as an optimization problem

1. Define "decision" variables
   → goal is to choose values for these.
2. Define objective function
3. Define constraints.

(1) Variables are easy: number of each coin
→ pennies, Nickels, Dimes, etc.
amount of each material used:
Cu, Ni, etc.

(2) Objective: maximize total dollar amount

maximize: 0.01 Pennies + 0.05 nickels + 0.1 Dimes + 0.25 quarters + 1 Dollars

(3) Constraints.

A. capture mineral usage:

Cu = 0.06 Pennies + 3.8 Nickels + 2.1 Dimes + 5.2 Quarters + 7.2 Dollars

B. Capture available material

$Cu \leq 1000$

$Ni \leq 50$

$Zi \leq 50$

$Mn \leq 50$

C. Note that we can't have a fraction of a penny: Need to indicate that they are integers

important for later

Integers
Pennies . Nickels Dimes Quarters Dollars

Note that LP allows variables to be real values. ILP only lets variables be integers.

3) Machine Example (LP)

company makes ~~two products~~ apple juice and orange juice (X, Y) using two machines (A, B)

Making one X takes 50 min of A and 30 min of B

Each Y takes 24 min on A and 33 min of B

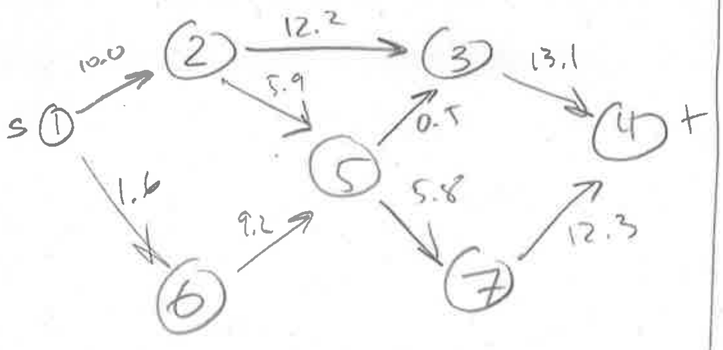
In one week, you can use A for 40 hours and B for 35 hours

Xs cost \$5 an Ys cost 2\$

$$\max\ 5x + 2y$$
s.t.
$$50x + 24y \leq 40$$
$$30x + 33y \leq 35$$

Optimization Pg 3

3. Max-Flow



1. what are the variables?
   - the amount of flow on each edge

   9 edges:
   f_1_2, f_2_3, etc.

2. What is the objective?

   Max f_3_4 + f_7_4

3. What are the constraints?

   i. capacity

   $f_{ij} \le c_{ij}$

   ii. conservation:

   $$\sum_{\forall i \mid \exists ij} f_{ij} = \sum_{\forall k \mid \exists jk} f_{jk}$$

   iii Non-negative

   $f_{i,j} \ge 0$

III ILP

LP allows variables to be real

ILP forces integer

see code example

4. MCF

See example

1. What are the variables?
   - amount of flow on each edge for each s,t pair

2. What is the objective?

   minimize congestion

   Demand[i,j] / Capacity(e)

   Z

3. What are the constraints?

   - conservation → like before plus the source and sink

Capacity:

Let's assume all demands are capacities are 1

Capacity constraint sums variables that "use an edge" subtracts Z, asserts less than or equal to 0.

Since sum of edges is congestion, this asserts that Z greater than or equal to congestion on one edge of the graph.

Since there is one constraint for each edge, Z will be the "max" congestion.

The LP will minimize the max congestion.

Optimization Pg 4

- If demands or capacities not equal to 1, the constraints must have coefficients to reflect that

want to assert that Z is greater than or equal to the weighted sum, where the weight reflects flow on an edge divided by capacity.

for source i destination j using edge e

Demand[i,j] / capacity[e]

Example: