MATH 464 – Lecture 3 (01/17/2023)

Today: * LP formulations

Download AMPL — follow instructions in Email.

LP Formulations.

→ Chapter 2 in AMPL book (ampl.com)

1. Blending problems – e.g., diet problem

The cost per package of different dinner packages, and the percentage daily value of nutrients (vitamins A, C, B1, B2) per package are given below. The problem is to find the combination of dinner packages to buy so that the requirement of each nutrient for a week are satisfied, i.e., get 700% of each nutrient, and minimize the total cost.

cost per package of dinners:

BEEF	beef	\$3.19
CHK	chicken	2.59
FISH	fish	2.29
HAM	ham	2.89
MCH	macaroni & cheese	1.89
MTL	meat loaf	1.99
SPG	spaghetti	1.99
TUR	turkey	2.49

Percentage daily values of nutrients per dinner package

	A	C	B1	B2
BEEF	60%	20%	10%	15%
CHK	8	0	20	20
FISH	8	10	15	10
HAM	40	40	35	10
MCH	15	35	15	15
MTL	70	30	15	15
SPG	25	50	25	15
TUR	60	20	15	10

Decision variables (d.v.'s)

Let x_j = # dinner packages of type j, j = BEEF, CHK, ..., TUR

Or, you could say, x_j = # dinner ..., for $j = 1, \dots, 8$, $1 \equiv$ BEEF, ..., $8 \equiv$ TUR.

It is important to define the d.v.'s clearly. In particular, statements of the form "Let x_{BEEF} = BEEF" would not cut it!

Objective Function

it's fine to use dots in this manner

$$\min \quad 3.19 X_{BEEF} + 2.59 X_{CHK} + \cdots + 2.49 X_{TUR} \qquad \text{(total cost)}$$

minimize

Constraints

Weekly requirement = 7x100

"subject to"

$$\text{s.t.} \quad 60 X_{BEEF} + 8 X_{CHK} + \cdots + 60 X_{TUR} \geq 700 \quad \text{(weekly Vit A)}$$

$$20 X_{BEEF} + \cdots + 20 X_{TUR} \geq 700 \quad \text{(Weekly Vit C)}$$

$$10 X_{BEEF} + \cdots + 15 X_{TUR} \geq 700 \quad \text{(weekly Vit B1)}$$

$$15 X_{BEEF} + \cdots + 10 X_{TUR} \geq 700 \quad \text{(" " B2)}$$

Sign restrictions

$$x_j \geq 0 \ \forall j \qquad \text{(non-neg)}$$

Here is the entire LP:

$$
\begin{array}{lll}
\min & 3.19 X_{BEEF} + 2.59 X_{CHK} + \cdots + 2.49 X_{TUR} & \text{(total cost)} \\
\text{s.t.} & 60 X_{BEEF} + 8 X_{CHK} + \cdots + 60 X_{TUR} \geq 700 & \text{(weekly Vit A)} \\
& 20 X_{BEEF} + \cdots + 20 X_{TUR} \geq 700 & \text{(" " C)} \\
& 10 X_{BEEF} + \cdots + 15 X_{TUR} \geq 700 & \text{(B1)} \\
& 15 X_{BEEF} + \cdots + 10 X_{TUR} \geq 700 & \text{(" " B2)} \\
& x_j \geq 0 \ \forall j & \text{(non-neg)}
\end{array}
$$

As we get more familiar and comfortable with LP formulations, we can jump directly to writing the final LP, and skip writing many of the intermediate steps in detail.

We solve the problem in AMPL. The model and data files are available from ampl.com (they're more general than what we need, as they consider some variations).

Model file:

```
set NUTR;
set FOOD;

param cost {FOOD} > 0;
param f_min {FOOD} >= 0;
param f_max {j in FOOD} >= f_min[j];

param n_min {NUTR} >= 0;
param n_max {i in NUTR} >= n_min[i];

param amt {NUTR,FOOD} >= 0;

var Buy {j in FOOD} >= f_min[j], <= f_max[j];

minimize Total_Cost:  sum {j in FOOD} cost[j] * Buy[j];

subject to Diet {i in NUTR}:
   n_min[i] <= sum {j in FOOD} amt[i,j] * Buy[j] <= n_max[i];
```

Data file (the actual numbers):

```
set NUTR := A B1 B2 C ;
set FOOD := BEEF CHK FISH HAM MCH MTL SPG TUR ;

param:   cost  f_min  f_max :=
  BEEF   3.19    0     100
  CHK    2.59    0     100
  FISH   2.29    0     100
  HAM    2.89    0     100
  MCH    1.89    0     100
  MTL    1.99    0     100
  SPG    1.99    0     100
  TUR    2.49    0     100 ;

param:   n_min  n_max :=
   A      700   10000
   C      700   10000
   B1     700   10000
   B2     700   10000 ;

param amt (tr):
          A    C   B1   B2 :=
   BEEF   60   20   10   15
   CHK     8    0   20   20
   FISH    8   10   15   10
   HAM    40   40   35   10
   MCH    15   35   15   15
   MTL    70   30   15   15
   SPG    25   50   25   15
   TUR    60   20   15   10 ;
```

AMPL session:

```
ampl: option solver cplex;
ampl: model "c:/Program
Files/AMPL/ampl_mswin64/models/diet.mod";
ampl: data "c:/Program Files/AMPL/ampl_mswin64/models/diet.dat";
ampl: solve;
CPLEX 20.1.0.0: optimal solution; objective 88.2
1 dual simplex iterations (0 in phase I)

ampl: display Buy;
Buy [*] :=
BEEF   0
 CHK   0
FISH   0
 HAM   0
 MCH  46.6667
 MTL   0
 SPG   0
 TUR   0
;

ampl:
```

It appears we want to buy just mac'n'cheese, as it's cheapest, and has enough of each of the four vitamins being studied!

We could consider several variations easily to this basic model.

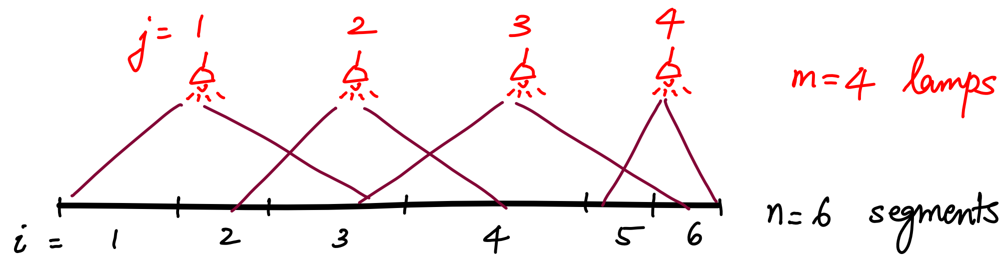
→ Bertsimas-Tsitsiklis: Intro to Linear Opt.

2. Lighting Problem (BT-ILO Prob 1.8):

d.v.s

Exercise 1.8 (Road lighting) Consider a road divided into n segments that is illuminated by m lamps. Let p_j be the power of the jth lamp. The illumination I_i (d.v.s) of the ith segment is assumed to be $\sum_{j=1}^{m} a_{ij}p_j$, where a_{ij} (data) are known coefficients. Let I_i^* (data) be the desired illumination of road i.

We are interested in choosing the lamp powers p_j so that the illuminations I_i are close to the desired illuminations I_i^* (data). Provide a reasonable linear programming formulation of this problem. Note that the wording of the problem is loose and there is more than one possible formulation.



The a_{ij} values (given data) captures all the information about whether lamp j illuminates segment i, and by how much. For instance, lamp 4 in the illustration above does not illuminate segment 2 at all, and hence $a_{24}=0$. Similarly, $a_{41}=0$ as well.

The main d.v.'s are specified in the problem itself — p_j and I_i. We will add some more d.v.'s in order to write the formulation(s).

Let p_j = power of lamp j, $j=1,..,m$

I_i = total illumination of road segment i, $i=1,\dots,n$.

Interpretation 1

* Illumination I_i must be at least I_i^*;
* Given the above condition is met, we want to minimize excess illumination, or
* minimize total power spent.

Let e_i = excess illumination in Segment i $(e_i = I_i - I_i^*)$ → extra d.v.'s

$$\min \sum_{i=1}^{n} e_i \qquad \text{(total excess illumination)}$$

or

$$\min \sum_{j=1}^{m} p_j \qquad \text{(total power)}$$

s.t.

$$I_i \geq I_i^*, \quad i=1,\dots,n \quad \text{(min. required illum.)}$$

$$I_i = \sum_{j=1}^{m} a_{ij} p_j, \quad i=1,\dots,n \quad \text{(illumination of segment } i\text{)}$$

$$e_i = I_i - I_i^*, \quad i=1,\dots,n \quad \text{(excess illumination)}$$

$$\text{all vars} \geq 0 \qquad \text{(non-neg)}$$

$e_i \geq 0 \Rightarrow I_i - I_i^* \geq 0$

$e_i \geq 0 \Rightarrow I_i \geq I_i^*$

So we could avoid the first set of constraints. But it's better to leave them in for improved readability.

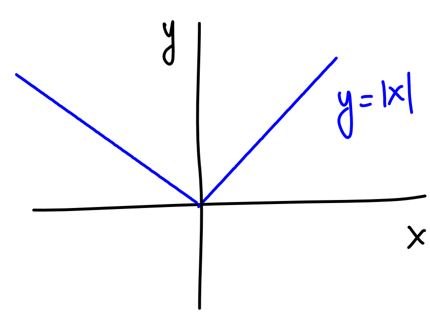
Interpretation 2: Get I_i "as close as possible" to I_i^*.

In other words, we want to minimize sum of $|I_i - I_i^*|$ $\forall i$.

absolute value

$y = |x|$ is nonlinear, but is "quite close to being linear"

$|x|$ is a piecewise linear (PL) function → more on modeling PL functions later.



3. Prob 1.11 (pg 36 BT-ILO): Currency conversion

Exercise 1.11 (Optimal currency conversion) Suppose that there are ~~N~~ n available currencies, and assume that one unit of currency i can be exchanged for r_{ij} units of currency j. (Naturally, we assume that $r_{ij} > 0$.) There also certain regulations that impose a limit u_i on the total amount of currency i that can be exchanged on any given day. Suppose that we start with ~~B~~ b units of currency 1 and that we would like to maximize the number of units of currency ~~N~~ n that we end up with at the end of the day, through a sequence of currency transactions. Provide a linear programming formulation of this problem. Assume that for any sequence $i_1, \ldots, i_k$ of currencies, we have $r_{i_1 i_2} r_{i_2 i_3} \cdots r_{i_{k-1} i_k} r_{i_k i_1} \le 1$, which means that wealth cannot be multiplied by going through a cycle of currencies.

We prefer to use n, b, etc. instead of N, B: it's just our notation convention!

n currencies

r_{ij} exchange rates

u_i max amount of currency i that can be exchanged

b amount of currency 1 at start

Assumption about r_{ij}'s: Cannot cycle through currencies to make money!

e.g. $\$ \longrightarrow ¥ \longrightarrow € \longrightarrow \$$

yen Euro

$r_{\$¥} = 100$ $\qquad 1\$ \longrightarrow 100¥ \longrightarrow 5€ \longrightarrow 12.5\$$ ✗

$r_{¥€} = \cancel{0.05}$ 0.005 $\qquad \longrightarrow 0.5€ \longrightarrow 0.75\$$ ✓

$r_{€\$} = \cancel{2.5}$ 1.5

Because of this assumption, it is clear that we will exchange any currency i to currency j only once in a day. If we convert a currency i to j more than once in a day, we will lose — might as well make all $i \to j$ conversions in one go!

We must do ~~some~~ exchanges for sure, as we want to convert (up to) b units of Currency 1 to as many units of Currency n as possible.

Let x_{ij} = # units of currency i exchanged to currency j

$i \neq j$, $i \in \{1,2,\dots,n\}$, $j \in \{1,2,\dots,n\}$.

Alternatively, we could restrict the index set of i (first index) to $\{1,2,\dots,n-1\}$ — we want to maximize the amount of currency n in the end, so an optimal solution would set $x_{nj} = 0 \; \forall j$ any way!

We'll finish the formulation in the next class...