Problem presented in Recitation class 1

Mithun Chakraborty

Problem source: `http://www.cs.unc.edu/~verma/comp116/L10/nutrition.html`

Problem description: Nutritional facts and prices (in appropriate units) per unit quantity of four food items are presented in the table below. Compute the quantities (non-integral values are allowed) of burgers, pizza slices, subs, and cups of cereals that should be consumed daily in order to minimize one's daily sodium intake without violating any of the following constraints:

- The daily protein intake should be at least 58 units.
- The daily fat intake should not exceed 65 units.
- The daily carb intake should not exceed 200 units.
- The daily calorie intake should not exceed 2000.
- The total cost of food per day should not exceed $30.

	Protein	**Fat**	**Carbs**	**Calories**	**Sodium**	**Price ($)**
1 **Burger**	25	42	57	700	1500	7
1 **Pizza slice**	9	9	26	210	600	4
1 **Sub**	20	37	88	780	1060	6
1 **Cup of cereals**	2	2	25	160	180	2

Linear Programming Formulation: Let x_B, x_P, x_S, x_C denote the required quantities of burgers, pizza slices, subs, and cups of cereals respectively. Then the problem reduces to:

$$\textbf{min} \quad 1500x_B + 600x_P + 1060x_S + 180x_C \qquad \text{(Sodium intake)}$$

$$\begin{array}{llcll}
\text{s.t.} & \mathbf{25x_B + 9x_P + 20x_S + 2x_C} & \geq & \mathbf{58}, & \textbf{(Protein constraint)} \\
 & 42x_B + 9x_P + 37x_S + 2x_C & \leq & 65, & \text{(Fat constraint)} \\
 & 57x_B + 26x_P + 88x_S + 25x_C & \leq & 200, & \text{(Carb constraint)} \\
 & 700x_B + 210x_P + 780x_S + 160x_C & \leq & 2000, & \text{(Calorie constraint)} \\
 & 7x_B + 4x_P + 6x_S + 2x_C & \leq & 30. & \text{(Cost constraint)}
\end{array}$$

Comments: Note the differences between the above formulation and a linear programming problem in its canonical form as discussed in class: here we have a minimization problem (instead of maximization), and the first constraint is of the "greater than or equal to" type (rather than a "less than or equal to" constraint). Our problem can be easily recast into the canonical form by simply flipping signs, *i.e.* changing the objective and the first constraint to

$$\mathbf{max} \quad -1500x_B - 600x_P - 1060x_S - 180x_C \qquad \text{(Sodium intake)}$$

$$\text{s.t.} \quad \mathbf{-25x_B - 9x_P - 20x_S - 2x_C \leq -58}, \qquad \textbf{(Protein constraint)}$$

$$\dots$$

This might be necessary if one is using, say, MATLAB to solve an LP (see `http://www.mathworks.com/help/optim/ug/linprog.html` for a documentation on the *linprog* function). However, the problem can be fed as it is into GLPK, as shown in the attached files.

- The file *rec1.mod* describes the problem in the canonical form. The generic model description and data for a particular instantiation are presented in the same file. The command-line instruction to solve this and store the output in a text file *results.txt* (attached) is

  ```
  glpsol --model rec1.mod --output results.txt
  ```

- The files *rec1_n.mod* and *rec1_n.dat* present a more preferable way of encoding an LP in GLPK: *rec1_n.mod* contains a straightforward description of the problem in its general form (without using any tricks like throwing it into the canonical form); the data file *rec1_n.dat* is separate and, if needed, can be generated using some other code in any language. The output is saved in *results_n.txt*, and the relevant instruction is

  ```
  glpsol --model rec1_n.mod --data rec1_n.dat --output results_n.txt
  ```

- If one is interested in looking for integer or binary solutions, it is very easy to do that in GLPK. Just put the word *integer* or *binary* in the description when you define your variable(s) in the *var* statement. An example is given in *rec1_integer.mod* (note that the only difference with *rec1.mod* is in the statement starting with *var*). The instruction for solving it is the same as before, and the output is saved in *results_integer.txt*.

  ```
  glpsol --model rec1_integer.mod --output results_integer.txt
  ```

- For more information on how to use *glpsol*, type

  ```
  glpsol --help
  ```