

The Diet Problem – McDonald's

Recall the diet problem assigned last week. On the course website, there is a MathProg model file for the linear program that solves the diet problem. In addition, there is a MathProg data file, consisting of the nutritional information for items sold at McDonald's.

1. Read through the model file and make sure you understand the linear program written in that file. (The solutions to last week's assignment may be helpful.)
2. Read through the data file, to get an idea of what the McDonald's data looks like. Read the comments carefully – there are some useful MathProg constructs in this file that we have not used before.

In addition, there is a parameter named `type` defined at the bottom of this data file, which is “commented out.” Leave it like this for now.

3. Solve the diet problem using the McDonald's data. The optimal total cost of the diet should be approximately \$5.36, and the optimal diet should consist of approximately 2.05 Cheeseburgers, 4.12 Sweet 'N Sour Sauces, 15.77 Honeys, 0.03 Chunky Chicken Salads, 2.27 Cheerios, 1.78 1% Lowfat Milks, and 0.41 Orange Juices.

Aside from its fractionality (let's assume this is OK), this optimal solution might seem a little strange to you. In particular, the McDonald's employees might give you dirty looks if you ask for 16 packets of honey.

4. In the data file, “uncomment” the lines corresponding to the `type` parameter (lines 239–302). As you can see, this parameter categorizes the foods into main dishes, side dishes, condiments, and beverages.
5. Add constraints to the model file to make the optimal diet more reasonable (well, as reasonable as a McDonald's diet can be). In particular, add constraints to ensure:
 - There are at least 4 beverages in the diet (hydration is important!)
 - There are at least 2 main dishes in the diet
 - There are at most 5 condiments in the diet

First, you will need to declare the `type` parameter in the model file with the other input parameters, using the statement below. The `symbolic` keyword indicates that the `type` parameter is not numeric.

```
param type{j in F} symbolic;
```

Second, writing constraints for these 3 requirements will require the “ : ” notation we discussed in Lesson 13. For example, if we wanted to ensure we have exactly 2 side dishes in our diet, we would write

$$\sum_{j \in F : \text{type}_j = \text{Side}} x_j = 2$$

In MathProg, this can be written using the statement below. Note the quotation marks around `Side`.

```
sum{j in F : type[j] = "Side"} x[j] = 4;
```

6. Upload your new model file into a folder named diet in the top level of your SA305 submission folder on Google Drive.