PyomoGallery (/github/Pyomo/PyomoGallery/tree/master)
/ diet (/github/Pyomo/PyomoGallery/tree/master/diet)

The Diet Problem

Summary

The goal of the Diet Problem is to select foods that satisfy daily nutritional requirements at minimum cost. This problem can be formulated as a linear program, for which constraints limit the number of calories and the amount of vitamins, minerals, fats, sodium, and cholesterol in the diet. Danzig (1990) notes that the diet problem was motivated by the US Army's desire to minimize the cost of feeding GIs in the field while still providing a healthy diet.

Problem Statement

The Diet Problem can be formulated mathematically as a linear programming problem using the following model.

Sets

F = set of foods
N = set of nutrients

Parameters

c_i = cost per serving of food i, $\forall i \in F$
a_{ij} = amount of nutrient j in food i, $\forall i \in F, \forall j \in N$
$Nmin_j$ = minimum level of nutrient j, $\forall j \in N$
$Nmax_j$ = maximum level of nutrient j, $\forall j \in N$
V_i = the volume per serving of food i, $\forall i \in F$
$Vmax$ = maximum volume of food consumed

Variables

x_i = number of servings of food i to consume

Objective

Minimize the total cost of the food
$\min \sum_{i \in F} c_i x_i$

Constraints

Limit nutrient consumption for each nutrient $j \in N$.
$Nmin_j \leq \sum_{i \in F} a_{ij} x_i \leq Nmax_j, \forall j \in N$

Limit the volume of food consumed
$\sum_{i \in F} V_i x_i \leq Vmax$

Consumption lower bound
$x_i \geq 0, \forall i \in F$

Pyomo Formulation

We begin by importing the Pyomo package and creating a model object:

In [1]:

```
from pyomo.environ import *
infinity = float('inf')

model = AbstractModel()
```

The sets F and N are declared abstractly using the `Set` component:

In [2]:

```
# Foods
model.F = Set()
# Nutrients
model.N = Set()
```

Similarly, the model parameters are defined abstractly using the `Param` component:

In [3]:

```
# Cost of each food
model.c    = Param(model.F, within=PositiveReals)
# Amount of nutrient in each food
model.a    = Param(model.F, model.N, within=NonNegativeReals)
# Lower and upper bound on each nutrient
model.Nmin = Param(model.N, within=NonNegativeReals, default=0.0)
model.Nmax = Param(model.N, within=NonNegativeReals, default=infinity)
# Volume per serving of food
model.V    = Param(model.F, within=PositiveReals)
# Maximum volume of food consumed
model.Vmax = Param(within=PositiveReals)
```

The `within` option is used in these parameter declarations to define expected properties of the parameters. This information is used to perform error checks on the data that is used to initialize the parameter components.

The `Var` component is used to define the decision variables:

In [4]:

```
# Number of servings consumed of each food
model.x = Var(model.F, within=NonNegativeIntegers)
```

The `within` option is used to restrict the domain of the decision variables to the non-negative reals. This eliminates the need for explicit bound constraints for variables.

The `Objective` component is used to define the cost objective. This component uses a rule function to construct the objective expression:

In [5]:

```
# Minimize the cost of food that is consumed
def cost_rule(model):
    return sum(model.c[i]*model.x[i] for i in model.F)
model.cost = Objective(rule=cost_rule)
```

Similarly, rule functions are used to define constraint expressions in the `Constraint` component:

In [6]:

```
# Limit nutrient consumption for each nutrient
def nutrient_rule(model, j):
    value = sum(model.a[i,j]*model.x[i] for i in model.F)
    return model.Nmin[j] <= value <= model.Nmax[j]
model.nutrient_limit = Constraint(model.N, rule=nutrient_rule)

# Limit the volume of food consumed
def volume_rule(model):
    return sum(model.V[i]*model.x[i] for i in model.F) <= model.Vmax
model.volume = Constraint(rule=volume_rule)
```

Putting these declarations all together gives the following model:

In [7]:

```
!cat diet.py
```

```
from pyomo.environ import *
infinity = float('inf')

model = AbstractModel()

# Foods
model.F = Set()
# Nutrients
model.N = Set()

# Cost of each food
model.c    = Param(model.F, within=PositiveReals)
# Amount of nutrient in each food
model.a    = Param(model.F, model.N, within=NonNegativeReals)
# Lower and upper bound on each nutrient
model.Nmin = Param(model.N, within=NonNegativeReals, default=0.0)
model.Nmax = Param(model.N, within=NonNegativeReals, default=infinity)
# Volume per serving of food
model.V    = Param(model.F, within=PositiveReals)
# Maximum volume of food consumed
model.Vmax = Param(within=PositiveReals)

# Number of servings consumed of each food
model.x = Var(model.F, within=NonNegativeIntegers)

# Minimize the cost of food that is consumed
def cost_rule(model):
    return sum(model.c[i]*model.x[i] for i in model.F)
model.cost = Objective(rule=cost_rule)

# Limit nutrient consumption for each nutrient
def nutrient_rule(model, j):
    value = sum(model.a[i,j]*model.x[i] for i in model.F)
    return model.Nmin[j] <= value <= model.Nmax[j]
model.nutrient_limit = Constraint(model.N, rule=nutrient_rule)

# Limit the volume of food consumed
def volume_rule(model):
    return sum(model.V[i]*model.x[i] for i in model.F) <= model.Vmax
model.volume = Constraint(rule=volume_rule)
```

Model Data

Since this is an abstract Pyomo model, the set and parameter values need to be provided to initialize the model. The following data command file provides a synthetic data set:

In [8]:

```
!cat diet.dat
```

```
param:  F:                          c      V  :=
  "Cheeseburger"                 1.84   4.0
  "Ham Sandwich"                 2.19   7.5
  "Hamburger"                    1.84   3.5
  "Fish Sandwich"                1.44   5.0
  "Chicken Sandwich"             2.29   7.3
  "Fries"                         .77   2.6
  "Sausage Biscuit"              1.29   4.1
  "Lowfat Milk"                   .60   8.0
  "Orange Juice"                  .72  12.0 ;

param Vmax := 75.0;

param:  N:        Nmin   Nmax :=
        Cal       2000      .
        Carbo      350    375
        Protein     55      .
        VitA       100      .
        VitC       100      .
        Calc       100      .
        Iron       100      . ;

param a:
                              Cal  Carbo Protein   VitA   VitC  Calc  Iron :=
  "Cheeseburger"              510     34      28     15      6    30    20
  "Ham Sandwich"              370     35      24     15     10    20    20
  "Hamburger"                 500     42      25      6      2    25    20
  "Fish Sandwich"             370     38      14      2      0    15    10
  "Chicken Sandwich"          400     42      31      8     15    15     8
  "Fries"                     220     26       3      0     15     0     2
  "Sausage Biscuit"           345     27      15      4      0    20    15
  "Lowfat Milk"               110     12       9     10      4    30     0
  "Orange Juice"               80     20       1      2    120     2     2 ;
```

Set data is defined with the `set` command, and parameter data is defined with the `param` command.

This data set considers the problem of designing a daily diet with only food from a fast food chain.

Solution

Pyomo includes a `pyomo` command that automates the construction and optimization of models. The GLPK solver can be used in this simple example:

In [9]:

```
!pyomo solve --solver=glpk diet.py diet.dat
```

```
[    0.00] Setting up Pyomo environment
[    0.00] Applying Pyomo preprocessing actions
[    0.00] Creating model
[    0.02] Applying solver
[    0.06] Processing results
    Number of solutions: 1
    Solution Information
      Gap: 0.0
      Status: optimal
      Function Value: 15.05
    Solver results file: results.json
[    0.06] Applying Pyomo postprocessing actions
[    0.06] Pyomo Finished
```

By default, the optimization results are stored in the file `results.yml`:

In [10]:

```
!cat results.yml
```

```
# ==========================================================
# = Solver Results                                         =
# ==========================================================
# ----------------------------------------------------------
#   Problem Information
# ----------------------------------------------------------
Problem:
- Name: unknown
  Lower bound: 15.05
  Upper bound: 15.05
  Number of objectives: 1
  Number of constraints: 10
  Number of variables: 10
  Number of nonzeros: 77
  Sense: minimize
# ----------------------------------------------------------
#   Solver Information
# ----------------------------------------------------------
Solver:
- Status: ok
  Termination condition: optimal
  Statistics:
    Branch and bound:
      Number of bounded subproblems: 89
      Number of created subproblems: 89
  Error rc: 0
  Time: 0.00977396965027
# ----------------------------------------------------------
#   Solution Information
# ----------------------------------------------------------
Solution:
- number of solutions: 1
  number of solutions displayed: 1
- Gap: 0.0
  Status: optimal
  Message: None
  Objective:
    cost:
      Value: 15.05
  Variable:
    x[Cheeseburger]:
      Value: 4
    x[Fries]:
      Value: 5
    x[Fish Sandwich]:
      Value: 1
    x[Lowfat Milk]:
      Value: 4
  Constraint: No values
```

This solution shows that for about $15 per day, a person can get by with 4 cheeseburgers, 5 fries, 1 fish sandwich and 4 milks.

References

- G.B. Dantzig. The Diet Problem, Interfaces 20(4), 1990, 43-47