problema-della-dieta-economica

July 10, 2021

1 problema della dieta economica

Si dispone di un certo numero di alimenti e di ognuno si conosce il costo per porzione e la composizione in termini di sostanze nutritive.

composizione degli alimenti

	pane	latte	uova	carne	dolce	req. minimi
calorie (cal)	110	160	180	260	420	2000
proteine (g)	4	8	13	14	4	50
calcio (mg)	2	285	54	80	22	700
costo × porzione	2	3	4	19	20	

Di ogni sostanza nutritiva è nota la quantità minima richiesta in un dato periodo.

Quali alimenti acquistare e in quali quantità in modo da garantire il fabbisogno nutrizionale del periodo e minimizzare il costo ?

2 modello parametrico

Parametri

- n numero di alimenti (ad es. pasta, carne, frutta, ...)
- m numero di sostanze nutritive (ad es. proteine, carboidrati, ….).

- a_{ji} q.tà di nutriente j contenuta in una porzione di alimento i
- c_i costo di una porzione di alimento i
- b_j q.tà minima di nutriente j richiesta per la sussistenza nel periodo dato

variabili decisionali

- $x_i \in \mathbb{R}$: numero di porzioni dell'alimento i che fanno parte della dieta

funzione obiettivo:
minimizzazione del costo totale

$$z = \min \sum_{i=1}^{n} c_i x_i$$

vincoli:

- Per ogni requisito nutrizionale occorre garantire l'assunzione della quantità minima

$$\sum_{i=1}^{n} a_{ji} x_i \geq b_j \qquad \forall j = 1, \ldots, m$$

- Il numero di porzioni di ogni alimento è una quantità non negativa

$$x_i \geq 0 \qquad \forall i = 1, \ldots, n$$

[23]:
```
from pyomo.environ import *
from pyomo.opt import SolverStatus, TerminationCondition

model = ConcreteModel()
model.dual = Suffix(direction=Suffix.IMPORT)

i = ['pane', 'latte', 'uova', 'carne', 'dolce']
c = [2, 3, 4, 19, 20]

j = ['calorie', 'proteine', 'calcio']
b = [2000, 50, 700]

a = [[110, 160, 180, 160, 420],
    [4, 8, 13, 14, 4],
    [2, 285, 54, 80, 22]]

model.i = Set(initialize=i)
model.j = Set(initialize=j)

a_dict = {}
for j, mj in enumerate(model.j):
    for i, mi in enumerate(model.i):
        a_dict[mj, mi] = a[j][i]

c = {mi: c[i] for i, mi in enumerate(model.i)}
b = {mj: b[j] for j, mj in enumerate(model.j)}

model.c = Param(model.i, initialize=c)
model.b = Param(model.j, initialize=b)

model.a = Param(model.j, model.i, initialize=a_dict)

model.x = Var(model.i, domain=NonNegativeReals, initialize=0)

obj_expr = sum(model.c[i]*model.x[i] for i in model.i)

model.cost = Objective(expr = obj_expr, sense=minimize)

model.constraints = ConstraintList()

for j in model.j:
    model.constraints.add(expr = sum(model.a[j, i]*model.x[i] for i in model.i)␣
 ↪>= model.b[j])

results = SolverFactory('glpk').solve(model)

model.display()
```

```
Model unknown

  Variables:
    x : Size=5, Index=i
        Key   : Lower : Value            : Upper : Fixed : Stale : Domain
        carne :     0 :              0.0 :  None : False : False : NonNegativeReals
        dolce :     0 :              0.0 :  None : False : False : NonNegativeReals
        latte :     0 : 2.35256203673864 :  None : False : False : NonNegativeReals
         pane :     0 : 14.7599097647438 :  None : False : False : NonNegativeReals
         uova :     0 :              0.0 :  None : False : False : NonNegativeReals

  Objectives:
    cost : Size=1, Index=None, Active=True
        Key  : Active : Value
        None :   True : 36.57750563970352

  Constraints:
    constraints : Size=3
        Key : Lower  : Body               : Upper
          1 : 2000.0 : 2000.0000000000002 :  None
          2 :   50.0 :  77.86013535288431 :  None
          3 :  700.0 :              700.0 :  None
```

[ ]: