problema di assegnamento

July 11, 2021

1 problema di assegnamento

Un sistema multiprocessore con 3 CPU (A,B e C) deve eseguire 3 processi (1,2 e 3).

Ogni processo richiede 10 ms di CPU time e può essere frazionato tra le CPU ma ogni CPU è disponibile per una quantità di tempo al massimo pari a 10 ms.

Date le differenti caratteristiche delle CPU, i costi unitari di processamento dipendono dalla coppia processo-CPU (vedi tabella).

Come assegnare i processi alle CPU in modo da minimizzare il costo totale?

CPU / processi	A	B	C
1	7	11	9
2	8	10	12
3	13	12	8

1.1 formulazione generale

Due insiemi A e B sono costituiti ognuno da n elementi.

Ogni elemento di A deve essere assegnato a un elemento di B (o frazionato tra più elementi di B);

a ogni elemento di B può essere assegnato al più un elemento di A (o frazioni di elementi di A che sommano 1).

Assegnare l'elemento $i \in A$ all'elemento $j \in B$ costa c_{ij}.

Qual è l'assegnamento di costo minimo?

1.2 modello parametrico

Parametri

- n numero di elementi di A e B (processi-CPU, persone-compiti, …)
- c_{ij} costo per assegnare l'elemento $i \in$ A all'elemento $j \in$ B

variabili decisionali

- $x_{ij} \in \mathbb{R}$: frazione di elemento $i \in$ A assegnato all'elemento $j \in$ B

funzione obiettivo:
minimizzazione del costo totale

$$z = \min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$$

vincoli:

- Ogni elemento di A deve essere assegnato completamente.

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \forall i = 1, \dots, n$$

- A ogni elemento di B può essere assegnata una quantità al più pari a 1

$$\sum_{i=1}^{n} x_{ij} \leq 1 \qquad \forall j = 1, \dots, n$$

- Le quantità assegnate sono non negative.

$$x_{ij} \geq 0 \qquad \forall i = 1, \dots, n \quad \forall j = 1, \dots, n$$

1.2.1 caso in cui gli elementi di A non siano frazionabili

- Se gli elementi di A (per es. i processi) non sono frazionabili allora occorre trasformare le variabili continue in variabili intere (e così diventano variabili logiche di *assegnamento*).

$$x_{ij} = \begin{cases} 1 \text{ se assegno l'elemento } i \in \text{A all'elemento } j \in \text{B} \\ 0 \text{ altrimenti} \end{cases}$$

cioè $x_{ij} = 1$ se nella tabella dei costi seleziono la casella nella riga i e colonna j

funzione obiettivo:
minimizzazione del costo totale

$$z = \min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$$

vincoli:

- Selezione <u>esattamente</u> un elemento per ogni riga della tabella dei costi

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \forall i = 1, \dots, n$$

- Selezione <u>esattamente</u> un elemento per ogni colonna della tabella dei costi

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad \forall j = 1, \dots, n$$

$$x_{ij} \in \{0,1\}$$

Il problema può essere descritto con un *grafo bipartito* in cui

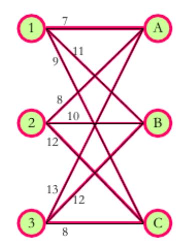
- i nodi rappresentano processi e CPU,
- l'arco $\{i, j\}$ descrive l'assegnamento del processo i alla CPU j (ed è quindi pesato con il corrispondente costo c_{ij} di processamento)

<u>Tabella dei costi</u>

CPU / processi	A	B	C
1	7	11	9
2	8	10	12
3	13	12	8

Quanti sono i possibili assegnamenti? $n!$

1. {A1, B2, C3} di costo 25
2. {A1, B3, C2} di costo 31
3. {A2, B1, C3} di costo 27
4. {A2, B3, C1} di costo 29
5. {A3, B2, C1} di costo 32
6. {A3, B1, C2} di costo 36



La soluzione ottima è l'assegnamento {A1,B2,C3} che costa 25

[6]:

```
from pyomo.environ import *
model = ConcreteModel('problema di trasporto')

i = [ 'pr1', 'pr2', 'pr3' ]

j = [ 'cpuA', 'cpuB', 'cpuC' ]

c = [[ 7, 11,  9],
     [ 8, 10, 12],
     [13, 12,  8]]

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

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

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

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

obj_expr = sum(sum(model.c[i, j]*model.x[i, j] for j in model.j) for i in model.i)
model.cost = Objective(expr = obj_expr, sense=minimize)

model.constraints = ConstraintList()
for i in model.i:
    model.constraints.add(expr = sum(model.x[i, j] for j in model.j) == 1)

for j in model.j:
    model.constraints.add(expr = sum(model.x[i, j] for i in model.i) == 1)

SolverFactory('glpk').solve(model)
model.display()
```

```
Model problema di trasporto

  Variables:
    x : Size=9, Index=x_index
        Key              : Lower : Value : Upper : Fixed : Stale : Domain
        ('pr1', 'cpuA') :     0 :   1.0 :     1 : False : False : Boolean
        ('pr1', 'cpuB') :     0 :   0.0 :     1 : False : False : Boolean
        ('pr1', 'cpuC') :     0 :   0.0 :     1 : False : False : Boolean
        ('pr2', 'cpuA') :     0 :   0.0 :     1 : False : False : Boolean
        ('pr2', 'cpuB') :     0 :   1.0 :     1 : False : False : Boolean
        ('pr2', 'cpuC') :     0 :   0.0 :     1 : False : False : Boolean
        ('pr3', 'cpuA') :     0 :   0.0 :     1 : False : False : Boolean
        ('pr3', 'cpuB') :     0 :   0.0 :     1 : False : False : Boolean
        ('pr3', 'cpuC') :     0 :   1.0 :     1 : False : False : Boolean

  Objectives:
    cost : Size=1, Index=None, Active=True
        Key  : Active : Value
        None :   True :  25.0

  Constraints:
    constraints : Size=6
        Key : Lower : Body : Upper
          1 :   1.0 :  1.0 :   1.0
          2 :   1.0 :  1.0 :   1.0
          3 :   1.0 :  1.0 :   1.0
          4 :   1.0 :  1.0 :   1.0
          5 :   1.0 :  1.0 :   1.0
          6 :   1.0 :  1.0 :   1.0
```

[ ]: