Mini-Project

Programmation Linéaire en nombres entiers

Guebli Ayoub Abdessami (G2)
Ghazi Salah Eddine (G2)

23 décembre 2023

1- Le problème peut être formulé comme un problème d'optimisation visant à minimiser le nombre total d'infirmiers nécessaires pour couvrir le service d'urgences tout au long de la semaine, en prenant en considération le cycle de travail de 4 jours consécutifs suivi de 3 jours de repos. Ainsi, pour chaque journée de la semaine, les infirmiers qui travaillent ce jour-là sont ceux qui ont commencé leur période de travail l'un des 4 jours précédents. Donc, on peut définir une liste de variables, chaque variable représentant le nombre d'infirmiers qui commencent le travail pour chaque jour.

Xi : Le nombre d'infirmiers commençant le jour i (pour i allant de 0 à 6, correspondant aux jours de la semaine(dimanche 0, lundi 1 …)).

La fonction objectif à minimiser est le nombre total d'infirmiers nécessaires :

Minimiser obj=∑Xi(pour i allant de 0 à 6)

On a la table qui affiche le nombre d'infirmiers nécessaires pour chaque jour de la semaine, donc les contraintes sont,

pour chaque jour i de 0 à 6 :

Xi−3 + X i−2 + X i−1 + Xi ≥ le nombre d'infirmiers nécessaires pour ce jour T[i]

Donc, en utilisant la bibliothèque Pyomo pour résoudre ce problème, nous exécutons le code suivant:

```python
from pyomo.environ import *

# Data
days = ['Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday']
demand = [12, 13, 10, 14, 11, 6, 8]

model = ConcreteModel()

# Variables
model.x = Var(range(7), within=NonNegativeIntegers)

# Objective
model.obj = Objective(expr=sum(model.x[day] for day in range(7)), sense=minimize, name="Total_Nurses_Cost")


# Constraints
model.Constraint = ConstraintList()
for i in range(7):
    constraint_expr = sum(model.x[j % 7] for j in range(i, i - 4, -1)) >= demand[i]
    model.add_component(f"Nurses_Coverage_Constraint_{days[i]}", Constraint(expr=constraint_expr))


# Solver
solver = SolverFactory('glpk')
solver.solve(model)

# Display results
print("Number of nurses who start work in each day:")
for i in range(7):
    print(f"{days[i]}: {model.x[i].value:.0f} nurses")

print("The number of nurses who work each day:")
for i in range(7):
    total_nurses_for_day = sum(model.x[j % 7].value for j in range(i, i - 4, -1))
    print(f"{days[i]}: {total_nurses_for_day:.0f} nurses")

print(f"Total number of nurses needed: {model.obj():.0f}")
print(f"Total expense: {model.obj() * 30000:.0f}")
```

Et ce code donne le résultat suivant :

```
Number of nurses who start work in each day:
Sunday: 7 nurses
Monday: 5 nurses
Tuesday: 0 nurses
Wednesday: 2 nurses
Thursday: 4 nurses
Friday: 2 nurses
Saturday: 0 nurses

The number of nurses who work each day:
Sunday: 13 nurses
Monday: 14 nurses
Tuesday: 12 nurses
Wednesday: 14 nurses
Thursday: 11 nurses
Friday: 8 nurses
Saturday: 8 nurses

Total number of nurses needed: 20
Total expense: 600000
```

Donc:

Le nombre total d'infirmiers est de 20.

La dépense globale est de 600 000 dinars par mois.

et l'effectif est comme suivant:

jour	Dimanche	Lundi	Mardi	Mercredi	Jeudi	Vendredi	Samedi
Les infirmières qui commencent ce jour	7	5	0	2	4	2	0
Les infirmières qui travaillent	13	14	12	14	11	8	8
Le nombre demandé	12	13	10	14	11	6	8

2- a) Le nombre d'infirmiers passe de 11 à 13 le jeudi:

en change "demand":

```python
from pyomo.environ import *

# Data
days = ['Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday']
demand = [12, 13, 10, 14, 13, 6, 8]

model = ConcreteModel()

# Variables
model.x = Var(range(7), within=NonNegativeIntegers)

# Objective
model.obj = Objective(expr=sum(model.x[day] for day in range(7)), sense=minimize, name="Total_Nurses_Cost")

# Constraints
model.Constraint = ConstraintList()
for i in range(7):
    constraint_expr = sum(model.x[j % 7] for j in range(i, i - 4, -1)) >= demand[i]
    model.add_component(f"Nurses_Coverage_Constraint_{days[i]}", Constraint(expr=constraint_expr))


# Solver
solver = SolverFactory('glpk')
solver.solve(model)

# Display results
print("Number of nurses who start work in each day:")
for i in range(7):
    print(f"{days[i]}: {model.x[i].value:.0f} nurses")

print("\nThe number of nurses who work each day:")
for i in range(7):
    total_nurses_for_day = sum(model.x[j % 7].value for j in range(i, i - 4, -1))
    print(f"{days[i]}: {total_nurses_for_day:.0f} nurses")

print(f"\nTotal number of nurses needed: {model.obj():.0f}")
print(f"Total expense: {model.obj() * 30000:.0f}")
```

Et ce code donne le résultat suivant :

```
Number of nurses who start work in each day:
Sunday: 6 nurses
Monday: 6 nurses
Tuesday: 0 nurses
Wednesday: 2 nurses
Thursday: 5 nurses
Friday: 1 nurses
Saturday: 0 nurses

The number of nurses who work each day:
Sunday: 12 nurses
Monday: 13 nurses
Tuesday: 12 nurses
Wednesday: 14 nurses
Thursday: 13 nurses
Friday: 8 nurses
Saturday: 8 nurses

Total number of nurses needed: 20
Total expense: 600000
```

Donc:

Le nombre total d'infirmiers est de 20.

La dépense globale est de 600 000 dinars par mois.

et l'effectif est comme suivant:

jour	Dimanche	Lundi	Mardi	Mercredi	Jeudi	Vendredi	Samedi
Les infirmières qui commencent ce jour	6	6	0	2	5	1	0
Les infirmières qui travaillent	12	13	12	14	14	8	8
Le nombre demandé	12	13	10	14	13	6	8

2- b) Le nombre d'infirmiers passe de 13 à 10 le lundi:

en change "demand":

```python
from pyomo.environ import *

# Data
days = ['Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday']
demand = [12, 10, 10, 14, 11, 6, 8]

model = ConcreteModel()

# Variables
model.x = Var(range(7), within=NonNegativeIntegers)

# Objective
model.obj = Objective(expr=sum(model.x[day] for day in range(7)), sense=minimize, name="Total_Nurses_Cost")

# Constraints
model.Constraint = ConstraintList()
for i in range(7):
    constraint_expr = sum(model.x[j % 7] for j in range(i, i - 4, -1)) >= demand[i]
    model.add_component(f"Nurses_Coverage_Constraint_{days[i]}", Constraint(expr=constraint_expr))


# Solver
solver = SolverFactory('glpk')
solver.solve(model)

# Display results
print("Number of nurses who start work in each day:")
for i in range(7):
    print(f"{days[i]}: {model.x[i].value:.0f} nurses")

print("\nThe number of nurses who work each day:")
for i in range(7):
    total_nurses_for_day = sum(model.x[j % 7].value for j in range(i, i - 4, -1))
    print(f"{days[i]}: {total_nurses_for_day:.0f} nurses")

print(f"\nTotal number of nurses needed: {model.obj():.0f}")
print(f"Total expense: {model.obj() * 30000:.0f}")
```

Et ce code donne le résultat suivant :

```
Number of nurses who start work in each day:
Sunday: 8 nurses
Monday: 2 nurses
Tuesday: 1 nurses
Wednesday: 3 nurses
Thursday: 5 nurses
Friday: 0 nurses
Saturday: 0 nurses

The number of nurses who work each day:
Sunday: 13 nurses
Monday: 10 nurses
Tuesday: 11 nurses
Wednesday: 14 nurses
Thursday: 11 nurses
Friday: 9 nurses
Saturday: 8 nurses

Total number of nurses needed: 19
Total expense: 570000
```

Donc:

Le nombre total d'infirmiers est de 19.

La dépense globale est de 570 000 dinars par mois.

et l'effectif est comme suivant:

jour	Dimanche	Lundi	Mardi	Mercredi	Jeudi	Vendredi	Samedi
Les infirmières qui commencent ce jour	8	2	1	3	5	0	0
Les infirmières qui travaillent	13	10	11	14	11	9	8
Le nombre demandé	12	10	10	14	11	6	8

3- Dans ce nouveau scénario, des modifications ont été apportées aux rémunérations en fonction du travail le weekend. L'objectif demeure de minimiser la dépense globale. En considérant que la dépense globale est désormais calculée comme la somme du produit du nombre d'infirmiers travaillant le weekend par leur salaire (34000 dinars) et du produit du nombre d'infirmiers ne travaillant pas le weekend par leur salaire (30000 dinars), la nouvelle fonction objectif à minimiser devient la suivante : dépense globale = (nombre d'infirmiers travaillant le weekend) * 34000 + (nombre d'infirmiers ne travaillant pas le weekend) * 30000. L'optimisation visera donc à ajuster le nombre d'infirmiers pour atteindre une solution optimale tout en prenant en compte ces changements de rémunération.

Les variables et les contraintes restent les mêmes,

Xi : Le nombre d'infirmiers commençant le jour i (pour i allant de 0 à 6, correspondant aux jours de la semaine(dimanche 0, lundi 1 …)).

contraintes:

pour chaque jour i de 0 à 6 :

$$X_{i-3} + X_{i-2} + X_{i-1} + X_i \geq \text{le nombre d'infirmiers nécessaires pour ce jour } T[i]$$

La fonction objectif change pour minimiser la dépense globale, c'est-à-dire le coût total, en ajustant le nombre d'infirmiers travaillant le weekend et ceux ne travaillant pas le weekend.

Chaque cycle de travail pour un infirmier consiste en 4 jours consécutifs. Ainsi, les infirmiers qui travaillent le weekend sont ceux qui commencent leur période de travail à l'un des jours suivants : mardi (indice 2), mercredi (3), jeudi (4), vendredi (5), ou samedi (6), et ils recevront un salaire de 34 000 dinars. Les autres infirmiers , qui ne travaillent pas le weekend (qui commencent dimanche ou lundi), recevront un salaire de 30 000 dinars.

donc l'objectif est

Minimiser obj=∑Xi(pour i allant de 2 à 6) * 34000 + ∑Xi(pour i allant de 0 à 1) * 30000

```python
from pyomo.environ import *

# Data
days = ['Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday']
demand = [12, 13, 10, 14, 11, 6, 8]

model = ConcreteModel()

# Variables
model.x = Var(range(7), within=NonNegativeIntegers)

# Objective
model.obj = Objective(  expr=sum(model.x[day] for day in range(3,7))*34000
                      +sum(model.x[day] for day in range(3))*30000,
                      sense=minimize, name="Total_Nurses_Cost")

# Constraints
model.Constraint = ConstraintList()
for i in range(7):
    constraint_expr = sum(model.x[j % 7] for j in range(i, i - 4, -1)) >= demand[i]
    model.add_component(f"Nurses_Coverage_Constraint_{days[i]}", Constraint(expr=constraint_expr))


# Solver
solver = SolverFactory('glpk')
solver.solve(model)

# Display results
print("Number of nurses who start work in each day:")
s=0
for i in range(7):
    print(f"{days[i]}: {model.x[i].value:.0f} nurses")
    s+=model.x[i].value


print("\nThe number of nurses who work each day:")
for i in range(7):
    total_nurses_for_day = sum(model.x[j % 7].value for j in range(i, i - 4, -1))
    print(f"{days[i]}: {total_nurses_for_day:.0f} nurses")

print(f"\nTotal number of nurses needed: {s:.0f}")
print(f"number of nurses who don't work in weekend: {sum(model.x[i].value for i in range(2)):.0f}")
print(f"number of nurses who work in weekend: {sum(model.x[i].value for i in range(2,7)):.0f}")
print(f"Total expense: {model.obj():.0f}")
```

Et ce code donne le résultat suivant :

```
Number of nurses who start work in each day:
Sunday: 7 nurses
Monday: 5 nurses
Tuesday: 0 nurses
Wednesday: 3 nurses
Thursday: 3 nurses
Friday: 1 nurses
Saturday: 1 nurses

The number of nurses who work each day:
Sunday: 12 nurses
Monday: 14 nurses
Tuesday: 13 nurses
Wednesday: 15 nurses
Thursday: 11 nurses
Friday: 7 nurses
Saturday: 8 nurses

Total number of nurses needed: 20
number of nurses who don't work in weekend: 12
number of nurses who work in weekend: 8
Total expense: 632000
```

Donc:

ui il a un effet sur le code et la dépense globale ,

Le nombre total d'infirmiers rest 20, mais

mais 8 infirmiers qui travaillent le weekend, et 12 qui ne travaillent pas le weekend.

La dépense globale est de 632 000 dinars par mois.

et l'effectif est comme suivant:

jour	Dimanche	Lundi	Mardi	Mercredi	Jeudi	Vendredi	Samedi
Les infirmières qui commencent ce jour	7	5	0	3	3	1	1
Les infirmières qui travaillent	12	14	13	15	11	7	8
Le nombre demandé	12	13	10	14	11	6	8

4- Dans cette question, nous ne comprenons pas exactement comment le groupe de garde va travailler et comment ils prendront leur repos au cours de la semaine. Par conséquent, supposons que l'équipe d'infirmières qui est dans son dernier jour de travail (le 4ème jour) sera l'équipe qui restera en garde pour la nuit.

donc, les variables et l'objectif restent les mêmes,

Xi : Le nombre d'infirmiers commençant le jour i (pour i allant de 0 à 6, correspondant aux jours de la semaine(dimanche 0, lundi 1 …)).

l'objectif est

Minimiser obj=∑Xi(pour i allant de 2 à 6) * 34000 + ∑Xi(pour i allant de 0 à 1) * 30000

Les contraintes dans ce cas vont changer. Nous avons que ⅓ des infirmières resteront pour la garde de nuit. Comme nous le supposons, l'équipe d'infirmières qui est dans son dernier jour de travail (le 4ème jour) sera l'équipe qui restera en garde pour la nuit. Cette équipe a déjà travaillé les 3 jours précédents, et elles commencent leur travail avant 3 jours (i-3). Nous ajoutons donc une autre contrainte pour chaque jour:

pour chaque jour i de 0 à 6 :

Xi−3 + X i−2 + X i−1 + Xi ≥ le nombre d'infirmiers nécessaires pour ce jour T[i]

Xi−3 ≥ (le nombre d'infirmiers nécessaires pour ce jour T[i]) /3

```python
from pyomo.environ import *

# Data
days = ['Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday']
demand = [12, 13, 10, 14, 11, 6, 8]

model = ConcreteModel()

# Variables
model.x = Var(range(7), within=NonNegativeIntegers)


# Objective
model.obj = Objective(  expr=sum(model.x[day] for day in range(3,7))*34000
                        +sum(model.x[day] for day in range(3))*30000,
                      sense=minimize, name="Total_Nurses_Cost")

# Constraints
model.Constraint = ConstraintList()
for i in range(7):
    constraint_expr = sum(model.x[j % 7] for j in range(i, i - 4, -1)) >= demand[i]
    model.add_component(f"Nurses_Coverage_Constraint_{days[i]}", Constraint(expr=constraint_expr))
    constraint_expr2 = model.x[(i-3) % 7] >= demand[i]/3
    model.add_component(f"Nurses_Garde_Constraint_{days[i]}", Constraint(expr=constraint_expr2))

# Solver
solver = SolverFactory('glpk')
solver.solve(model)

# Display results
print("Number of nurses who start work in each day:")
s=0
for i in range(7):
    print(f"{days[i]}: {model.x[i].value:.0f} nurses")
    s+=model.x[i].value

print("\nThe number of nurses who work each day:")
for i in range(7):
    total_nurses_for_day = sum(model.x[j % 7].value for j in range(i, i - 4, -1))
    print(f"{days[i]}: {total_nurses_for_day:.0f} nurses")

print(f"\nTotal number of nurses needed: {s:.0f}")
print(f"number of nurses who don't work in weekend: {sum(model.x[i].value for i in range(2)):.0f}")
print(f"number of nurses who work in weekend: {sum(model.x[i].value for i in range(2,7)):.0f}")
print(f"Total expense: {model.obj():.0f}")
```

la résultat:

```
Number of nurses who start work in each day:
Sunday: 5 nurses
Monday: 4 nurses
Tuesday: 2 nurses
Wednesday: 3 nurses
Thursday: 4 nurses
Friday: 5 nurses
Saturday: 4 nurses

The number of nurses who work each day:
Sunday: 18 nurses
Monday: 18 nurses
Tuesday: 15 nurses
Wednesday: 14 nurses
Thursday: 13 nurses
Friday: 14 nurses
Saturday: 16 nurses

Total number of nurses needed: 27
number of nurses who don't work in weekend: 9
number of nurses who work in weekend: 18
Total expense: 874000
```

Donc:

Le nombre total d'infirmiers est de 27.

La dépense globale est de 874 000 dinars par mois.

et l’effectif est comme suivant:

jour	Dimanche	Lundi	Mardi	Mercredi	Jeudi	Vendredi	Samedi
Les infirmières qui commencent ce jour	5	4	2	3	4	5	4
Les infirmières qui travaillent	18	18	15	14	13	14	16
L'équipe de garde de nuit. (les infirmiers dans son 4eme jour de travaille)	4	5	4	5	4	2	3
Le nombre demandé	12	13	10	14	11	6	8