LINMA1702 modèles et méthodes d'optimisation : Logistique de la vaccination en Belgique

Groupe 6 : Théau Lepouttre, Eliott Van Dieren et Nicolas Mil-Homens Cavaco

2020-2021

0.1 Premier jet

0.1.1 Question 1.1

Soit les variables suivantes,

- $x \equiv$ le nombre de doses qui arrivent au centre de vaccination ;
- $y_i \equiv$ le nombre de doses administrées à la classe d'âge i ;

contraintes par les paramètres suivants,

- $b_l \equiv$ le nombre maximal de vaccins livrés par jour ;
- $b_v \equiv$ le nombre maximal de vaccins administrés par jour ;
- $c_t \equiv$ le coût de transport ;
- $c_v \equiv$ le coût de vaccination ;
- $c_{tot} \equiv$ le budget total disponible ;
- $n_s \equiv$ le nombre de personnes susceptibles ;
- $n_{nv} \equiv$ le nombre de personnes qui ne veulent pas se faire vacciner parmi les suceptibles.

Puisque dans un premier temps, on suppose que le stockage des vaccins est impossible, la totalité de vaccins livrés doivent être administrés ou doivent être jetés, c'est-à-dire que $x \geq y$.

Le problème s'écrit donc

$$
\begin{aligned}
\max_{x \in \mathbb{R}, y \in \mathbb{R}^m} \quad & \textstyle\sum_{i=1}^{m} y_i & & \\
x - \sum_{i=1}^{m} y_i \quad & \geq & & 0 \\
c_t\ x + c_v \sum_{i=1}^{m} y_i \quad & \leq & & c_{tot} \\
x \quad & \leq & & b_l \\
\sum_{i=1}^{m} y_i \quad & \leq & & b_v \\
y_i \quad & \leq & & (n_s)_i - (n_{nv})_i \\
x,\ y \quad & \geq & & 0
\end{aligned}
$$

```python
import numpy as np
import matplotlib.pyplot as plt
from mip import *

# Donnees du probleme
m     = 6    # nombre de classes d'age
c_t   = 1    # Prix de livraison d'un vaccin
c_v   = 1    # prix d'administration d'un vaccin
c_tot = 100  # budget total autorise
b_l   = 100  # nombre maximal de vaccins livres par jour
b_v   = 150  # nombre maximal de vaccins administres par jour

# nombre de personnes suceptibles dans chaque classe d'age
n_s   = np.ones(m) * 20
# nombre de personnes ne voulant pas etre vaccinees dans chaque classe d'age
n_nv  = np.ones(m) * 1

model = Model('centre unique', sense=MAXIMIZE, solver_name=CBC)

# Variables
x = model.add_var(var_type=INTEGER, lb=0)
y = np.array([model.add_var(var_type=INTEGER, lb=0) for i in range(m)])

# Objectif
model.objective = maximize(xsum(y))

# Contraintes
model += c_t * x + xsum(c_v * y) <= c_tot
model += x - xsum(y) >= 0
model += x <= b_l
model += xsum(y) <= b_v
for i in range(m):
    model += y[i] <= n_s[i] - n_nv[i]

model.optimize()

print(f"x = {x.x}")
print(f"y = {[y[i].x for i in range(m)]}")
print(f"f(x,y) = {model.objective_value}")
```