11/01/2022

Méthodes numériques et optimisation

Projet n°1 : Simplex

Enseignant lecteur : Thiago Abreu

Etudiant : Thomas BOUCHET

Lien GitHub : https://github.com/ThomasToto/Optimisation/tree/master/Projet

EPISEN – 2021/2022

Introduction :

Dans le cadre du cours de méthodes numériques et d'optimisation nous sommes menés à effectuer une IHM (Interface Homme Machine) visant à utiliser plus simplement l'algorithme du simplex. Nous verrons dans quelques instants que nous avons en réalité utilisé la méthode du Big M.

Définition des problèmes d'optimisation :

Question 1 :

Un fabricant produit trois types d'accessoires en plastique. Le temps nécessaire au moulage, à la découpe et à l'emballage est indiqué dans le Tableau 1. (Les temps sont indiqués en heures par douzaine d'accessoires).

Tableau 1: Coûts et profits associés à la production de produits en plastique.

Procédure / Produit	**Type A**	**Type B**	**Type C**	**Temps total disponible**
Moulage	1	2	3/2	12.000,00
Découpage	2/3	2/3	1	4.600,00
Emballage	1/2	1/3	1/2	2.400,00
Profit	11 €	16 €	15 €	

Combien de douzaines de chaque type d'accessoire doivent être produites pour obtenir un profit maximal ?

Après lecture du sujet de la question n°1 on peut définir le problème d'optimisation de la manière suivante :

On pose x1,x2 et x3 les quantités de chaque produit.
Nous avons la fonction objective suivante : MIN(Z) = 11x1 + 16x2 + 15x3
Puis nous avons la fonction objective sous les contraintes suivantes :

x1 + 2x2 + 3/2x3 <= 12000
2/3x1 + 2/3x2 + x3 <= 4600
1/2x1 +1/3x2 + 1/2x3 <= 2400

Question 2 :

Une petite compagnie pétrolière possède deux raffineries. La raffinerie 1 coûte 20 000 dollars par jour et peut produire chaque jour 400 barils de pétrole de qualité supérieure, 300 barils de pétrole de qualité moyenne et 200 barils de pétrole de qualité inférieure. La raffinerie 2 est plus récente et plus moderne. Son coût d'exploitation est de 25 000 dollars par jour et elle peut produire chaque jour 300 barils de pétrole de qualité supérieure, 400 barils de pétrole de qualité moyenne et 500 barils de pétrole de qualité inférieure.
L'entreprise a des commandes totalisant 25000 barils de pétrole de qualité supérieure, 27000 barils de pétrole de qualité moyenne et 30000 barils de pétrole de qualité inférieure. Combien de jours doit-elle faire fonctionner chaque raffinerie pour minimiser ses coûts tout en raffinant suffisamment de pétrole pour honorer ses commandes ?

Après lecture du sujet de la question n°2 on peut définir le problème d'optimisation de la manière suivante :

On pose x1 et x2 le nombre de journées des usines 1 et 2.
Nous avons la fonction objective suivante : MIN(Z) Z= 20000x1 + 25000x2
Or on cherche avoir une maximisation donc ceci donne : MAX(Z) Z= -20000x1 -25000x2
Puis nous avons la fonction objective sous les contraintes suivantes :

400x1 + 300x2 => 25000
300x1 + 400x2 => 27000
200x1 + 500x2 => 30000

Question 3 :

Une entreprise automobile possède deux usines. Une usine a 400 voitures (d'un certain modèle) en stock et l'autre usine a 300 voitures (du même modèle) en stock. Deux clients commandent ce modèle de voiture. Le premier client a besoin de 200 voitures, et le second de 300 voitures. Le coût de l'expédition des voitures des deux usines aux clients est indiqué dans le Tableau 2.

Tableau 2 : Coûts d'expédition de véhicules.

	Client 1	**Client 2**
Usine 1	30 €	25 €
Usine 2	36 €	30 €

Comment l'entreprise doit-elle expédier les voitures afin de minimiser les frais d'expédition ?

Après lecture du sujet de la question n°2 on peut définir le problème d'optimisation de la manière suivante :
On pose x1 et x2 les quantités envoyées des entreprises 1 et 2 au client 1. Et x3 et x4 quantités de voitures vers client 2.

On se retrouve avec la fonction objective suivante : MIN(Z) Z = 30x1 + 36x2 + 25x3 + 30x4
Sous les contraintes :

x1 + x2 = 200
x3 + x4 = 300
x1 + x3 <= 400
x2 + x4 <= 300

<u>La fonction simplex :</u>

La fonction simplex_BigM est le coeur de programme. C'est cette fonction qui gère le calcul des solutions d'optimisation via la fonction objective et les contraintes.

La fonction simplex_BigM prend en paramètre :

- coeffContrainte → les coefficients des contraintes (type : Liste de liste)

- resultat → résultat des contraintes (type : liste)

- coeffFonction → les coefficients de la fonction objective (type : liste)

- signe → le signe des contraintes, s'il s'agit d'une égalité, ou inégalité (type : liste)
  → -1 : '<=' , 1 : '>=' , 0 : '='

- maxOuMin → s'il s'agit d'une maximisation ou minimisation (type : String)
  → 'max' correspond à une maximisation, 'min' à une minimisation

- montrerIteration → permet de choisir d'afficher les tableaux ou non (type : int)
  → 1 = montrer les itérations, 0 ne pas montrer les itérations

Exemple d'appel de la fonction simplex_BigM :

```
coeffFonction = [ 2,1 ]
coeffContrainte = [[2,1], [1,1]]
resultat = [8, 5]
signe = [-1, 1]
simplex_BigM(coeffContrainte,resultat,coeffFonction,signe,'max',montrerIteration)
```

Cet appel correspond au problème d'optimisation suivant :

$$(P_1)\begin{cases} \max\ z = & 2x_1 + x_2 \\ 2x_1 + x_2 & \leq 8 \\ x_1 + x_2 & \geq 5 \\ x_1, x_2 & \geq 0 \end{cases}$$

Lien entre la fonction et l'IHM :

Le but de l'IHM était donc de récolter les données saisies par l'utilisateur, c'est à dire tous les paramètres nécessaires pour l'appel de la fonction simplex_BigM.
En plus de récolter les données l'IHM devait aussi reformater les données car la fonction acceptait principalement que des listes.

Zone de saisie de l'IHM :

L'utilisateur saisie des chaînes de caractères. Le principe est de séparer les différentes valeurs par des virgules. Ceci m'a permis par la suite de pouvoir formater les données plus simplement.

Pour le champ de saisie 1, la fonction attend une chaîne de caractère donc je n'ai pas effectué de formatage particulier sur la saisie de l'utilisateur.

Pour le champ de saisie 2, 4, 5 les valeurs doivent être séparées par des virgules pour que le programme les transforme en liste par la suite.

Exemple de formatage du champ 4 :

```
champ4 = self.lineSigneContrainte.text()
champ4 = champ4.split(',')
champ4 = [int(i) for i in champ4]
```

On récupère la valeur de le champ de saisie, on transforme les valeurs en une liste dans l'indice séparateur est la virgule puis on transforme chaque élément de la liste en entier.

Pour le champ 3 le formatage est différent car la fonction attendait une liste de liste. Le formatage est donc le suivant :

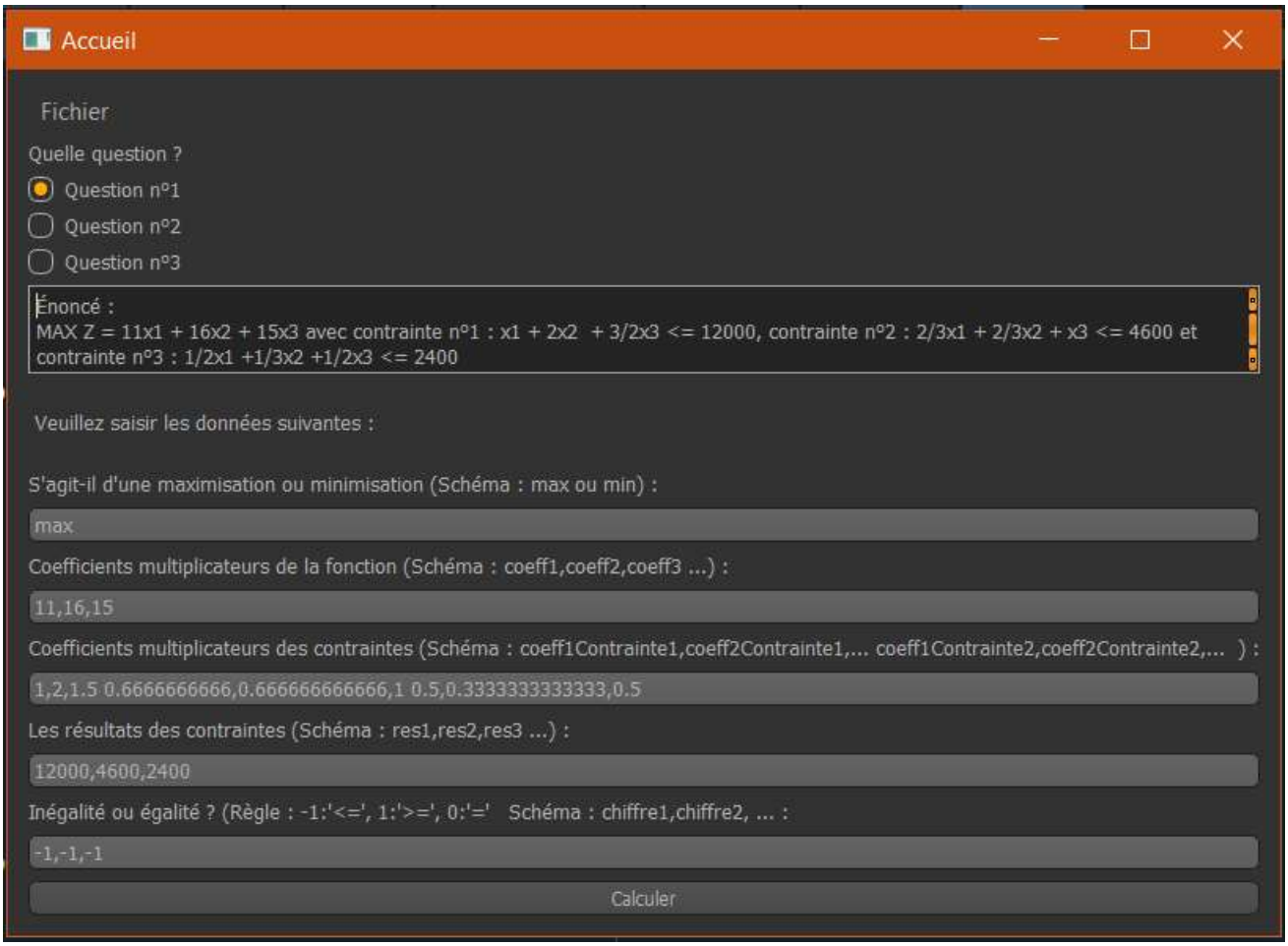
```
champ2 = self.lineCoeffMultiContrainte.text()
champ2 = champ2.split(' ')
champ2 = [i.split(',') for i in champ2]

for i in range(len(champ2)):
    for k in range(len(champ2[i])):
        champ2[i][k] = float(champ2[i][k])
print(champ2)
```

On récupère la valeur du champ de saisie puis on sépare les blocs de la saisie via l'indice espace. On enlève les virgules non désirées et on effectue le formatage liste de liste.

Les données étant formatées on peut maintenant les mettre en tant que paramètre de la fonction simplex_BigM pour l'appeler.

IHM Finale :



Structure du code :

Les différents fichiers / Structure du programme :
- Views.py → script s'occupant de l'IHM
- simplex_BigM → script s'occupant du calcul de Simplex
- style.css → s'occupe du style de l'IHM

Utilisation de l'IHM :

- Pour lancer l'IHM, il suffit de lancer le fichier views.py avec la commande python3 views.py.

- L'IHM va alors être lancée. Les questions 1,2,3 sont déjà enregistrées, il suffit de sélectionner la question souhaitée et de cliquer sur "Calculer".

- Le résultat apparaît alors dans la console. Le résultat possède les tableaux de chaque itérations et les solutions finales.

Conclusion :

Pour conclure, j’avais déjà effectuer la fonction BigM ce qui m’a permis de gagner du temps dans la réalisation de ce projet. Il m’a quand même fallu effectuer le lien entre la fonction et l’IHM. Ceci a été un peu casse tête car le formatage n’était pas simple. Néanmoins j’ai réussi à rendre l’IHM totalement fonctionnelle dans n’importe quel cas ce qui est positif.