Travaux Pratiques : Programmation Linéaire et Résolution Graphique avec Python

Votre Nom

22 octobre 2024

Table des matières

1 Introduction

La programmation linéaire est une méthode d'optimisation pour maximiser ou minimiser une fonction linéaire, appelée fonction objectif, sous des contraintes linéaires. Elle est largement utilisée dans divers domaines tels que l'économie, l'ingénierie et la logistique.

2 Objectifs du TP

— Comprendre les concepts fondamentaux de la programmation linéaire.
— Savoir formuler mathématiquement un problème d'optimisation linéaire.
— Apprendre à résoudre un problème de programmation linéaire de manière graphique.
— Utiliser Python pour résoudre des problèmes de programmation linéaire et visualiser les solutions.

3 Partie I : Formulation Mathématique

3.1 Exemple de Problème

Une entreprise fabrique deux produits, A et B. Chaque produit nécessite des ressources limitées :

— Le produit A rapporte un bénéfice de 40€ par unité.
— Le produit B rapporte un bénéfice de 30€ par unité.

Les contraintes de production sont les suivantes :

	Produit A	Produit B
Matière première (kg)	2	1
Temps de production (heures)	3	2

Les ressources disponibles sont :
— Matière première : 100 kg.
— Temps de production : 120 heures.

3.2 Formulation du Modèle

Définissons :
— x : nombre d'unités du produit A à produire.
— y : nombre d'unités du produit B à produire.

Fonction objectif à maximiser :

$$Z = 40x + 30y$$

Sous contraintes :

$$\begin{cases} 2x + y \leq 100 & \text{(Matière première)} \\ 3x + 2y \leq 120 & \text{(Temps de production)} \\ x \geq 0, \quad y \geq 0 & \text{(Non-négativité)} \end{cases}$$

4 Partie II : Résolution Graphique

4.1 Traçage des Contraintes

1. Représentez les droites des contraintes sur un graphique.
2. Identifiez la région faisable définie par les contraintes.

4.2 Détermination de la Solution Optimale

1. Tracez les isoprofits de la fonction objectif.
2. Trouvez le point optimal qui maximise Z dans la région faisable.

5 Partie III : Résolution avec Python

5.1 Installation des Bibliothèques

Assurez-vous que les bibliothèques `numpy`, `matplotlib`, `scipy` et `PuLP` sont installées :

```
pip install numpy matplotlib scipy pulp
```

5.2 Définition d'un Problème d'Optimisation en Python

Dans cette section, nous allons apprendre comment définir et résoudre un problème d'optimisation en Python en utilisant les bibliothèques `PuLP` et `scipy.optimize`.

5.2.1 Utilisation de `PuLP` pour la Programmation Linéaire

Étape 1 : Installation de PuLP Assurez-vous que la bibliothèque PuLP est installée :

```
pip install pulp
```

Étape 2 : Formulation du Problème Reprenons l'exemple présenté précédemment :

— Fonction objectif à maximiser :

$$Z = 40x + 30y$$

— Contraintes :

$$\begin{cases} 2x + y \leq 100 \\ 3x + 2y \leq 120 \\ x \geq 0, \quad y \geq 0 \end{cases}$$

Étape 3 : Écriture du Code Python avec PuLP

```python
import pulp

# 1. Cr ation du probl me de maximisation
prob = pulp.LpProblem("Maximiser_le_profit", pulp.LpMaximize)

# 2. D finition des variables de d cision
x = pulp.LpVariable('x', lowBound=0, cat='Continuous')
y = pulp.LpVariable('y', lowBound=0, cat='Continuous')

# 3. D finition de la fonction objectif
```

```
prob += 40 * x + 30 * y, "Profit_total"

# 4. Ajout des contraintes
prob += 2 * x + y <= 100, "Contrainte_de_matiere_premiere"
prob += 3 * x + 2 * y <= 120, "Contrainte_de_temps_de_production"

# 5. R solution du probl me
prob.solve()

# 6. Affichage des r sultats
print("Statut de la solution :", pulp.LpStatus[prob.status])
print(f"Quantit  optimale de produit A (x) : {x.varValue}")
print(f"Quantit  optimale de produit B (y) : {y.varValue}")
print(f"Profit total optimal : {pulp.value(prob.objective)}    ")
```

Explications :

- **Création du problème** : On crée un objet `LpProblem` en spécifiant qu'il s'agit d'un problème de maximisation.
- **Définition des variables de décision** : `x` et `y` sont les quantités des produits A et B.
- **Fonction objectif** : On ajoute la fonction objectif au problème.
- **Contraintes** : Les contraintes sont ajoutées de manière similaire à la fonction objectif.
- **Résolution** : La méthode `solve()` résout le problème en utilisant le solveur par défaut.
- **Résultats** : On affiche le statut de la solution et les valeurs optimales des variables.

Résultat attendu :

```
Statut de la solution : Optimal
Quantité optimale de produit A (x) : 20.0
Quantité optimale de produit B (y) : 60.0
Profit total optimal : 2800.0 €
```

5.2.2 Utilisation de `scipy.optimize` pour la Programmation Linéaire

Étape 1 : Importation de la fonction `linprog`

```
from scipy.optimize import linprog
```

Étape 2 : Formulation du Problème pour `linprog`

- Fonction objectif à minimiser (coefficients négatifs pour maximiser) :

$$c = [-40, -30]$$

- Matrice des contraintes :

$$A = \begin{bmatrix} 2 & 1 \\ 3 & 2 \end{bmatrix}$$

- Vecteur des bornes des contraintes :

$$b = \begin{bmatrix} 100 \\ 120 \end{bmatrix}$$

Étape 3 : Écriture du Code Python avec `linprog`

```python
from scipy.optimize import linprog

# Coefficients de la fonction objectif (n gatifs pour la maximisation)
c = [-40, -30]

# Matrice des coefficients des contraintes
A_ub = [
    [2, 1],
    [3, 2]
]

# Vecteur des bornes des contraintes
b_ub = [100, 120]

# Bornes des variables
x_bounds = (0, None)
y_bounds = (0, None)

# R solution du probl me
res = linprog(c, A_ub=A_ub, b_ub=b_ub, bounds=[x_bounds, y_bounds],
   method='highs')

# Affichage des r sultats
if res.success:
    print("Statut de la solution :", res.message)
    print(f"Quantit  optimale de produit A (x) : {res.x[0]}")
    print(f"Quantit  optimale de produit B (y) : {res.x[1]}")
    print(f"Profit total optimal : {-res.fun}    ")
else:
    print("La solution n'a pas   t   trouv e :", res.message)
```

Résultat attendu :

```
Statut de la solution : Optimization terminated successfully. (HiGHS Status 7: Optimal)
Quantité optimale de produit A (x) : 20.0
Quantité optimale de produit B (y) : 60.0
Profit total optimal : 2800.0 €
```

5.3 Conseils pour la Définition des Problèmes d'Optimisation

- **Formulation Mathématique** : Avant de coder, assurez-vous que votre problème est correctement formulé mathématiquement.
- **Choix de la Bibliothèque** : Utilisez `PuLP` pour la programmation linéaire, `scipy.optimize` pour des problèmes plus généraux.
- **Types de Contraintes** : Les contraintes peuvent être d'égalité ou d'inégalité.
- **Bornes des Variables** : Définissez les bornes pour vos variables si nécessaire.
- **Vérification des Résultats** : Vérifiez le statut de la solution pour vous assurer que l'optimisation a abouti.

5.4 Exemple Complet avec PuLP dans le Contexte du TP

Prenons l'Exercice 1 du TP.

5.4.1 Formulation Mathématique

Une ferme cultive des tomates et des concombres. Les tomates rapportent 3€ par kg et les concombres 2€ par kg. Les contraintes sont les suivantes :

— Terrain disponible : 50 hectares.
— Main-d'œuvre disponible : 1000 heures.
— Les tomates nécessitent 1 hectare et 20 heures de travail par kg.
— Les concombres nécessitent 1 hectare et 10 heures de travail par kg.

Définissons :

— T : quantité de tomates (en kg) à cultiver.
— C : quantité de concombres (en kg) à cultiver.

Fonction objectif à maximiser :

$$Z = 3T + 2C$$

Sous contraintes :

$$\begin{cases} T + C \leq 50 \quad \text{(Terrain)} \\ 20T + 10C \leq 1000 \quad \text{(Main-d'œuvre)} \\ T \geq 0, \quad C \geq 0 \quad \text{(Non-négativité)} \end{cases}$$

5.4.2 Code Python avec PuLP

```python
import pulp

# Cr ation du probl me de maximisation
prob = pulp.LpProblem("Maximiser_le_profit_de_la_ferme", pulp.
    LpMaximize)

# D finition des variables de d cision
T = pulp.LpVariable('Tomates', lowBound=0, cat='Continuous')
C = pulp.LpVariable('Concombres', lowBound=0, cat='Continuous')

# Fonction objectif
prob += 3 * T + 2 * C, "Profit_total"

# Contraintes
prob += T + C <= 50, "Contrainte_de_terrain"
prob += 20 * T + 10 * C <= 1000, "Contrainte_de_main_d'oeuvre"

# R solution du probl me
prob.solve()

# Affichage des r sultats
print("Statut de la solution :", pulp.LpStatus[prob.status])
print(f"Quantit  optimale de tomates (kg) : {T.varValue}")
print(f"Quantit  optimale de concombres (kg) : {C.varValue}")
print(f"Profit total optimal : {pulp.value(prob.objective)}    ")
```

Résultat attendu :

```
Statut de la solution : Optimal
Quantité optimale de tomates (kg) : 30.0
Quantité optimale de concombres (kg) : 20.0
Profit total optimal : 130.0 €
```

6 Exercices

6.1 Exercice 1

Une ferme cultive des tomates et des concombres. Les tomates rapportent 3€ par kg et les concombres 2€ par kg. Les contraintes sont les suivantes :

— Terrain disponible : 50 hectares.
— Main-d'œuvre disponible : 1000 heures.
— Les tomates nécessitent 1 hectare et 20 heures de travail par kg.
— Les concombres nécessitent 1 hectare et 10 heures de travail par kg.

Tâches :

1. Formulez le problème en termes de programmation linéaire.
2. Résolvez-le graphiquement.
3. Vérifiez la solution avec Python.

6.2 Exercice 2

Une usine produit des tables et des chaises. Chaque table nécessite 4 unités de bois et 2 heures de travail. Chaque chaise nécessite 3 unités de bois et 1 heure de travail. Les bénéfices sont de 70€ par table et 50€ par chaise. Les ressources disponibles sont 240 unités de bois et 100 heures de travail.

Tâches :

1. Formulez le problème en programmation linéaire.
2. Utilisez Python pour trouver la solution optimale.

6.3 Exercice 3

Une entreprise fabrique deux types de chaussures : sport et classique. Les chaussures de sport rapportent 50€ par paire, et les classiques 40€ par paire. Les contraintes de production sont les suivantes :

— Temps de couture disponible : 800 heures.
— Temps d'assemblage disponible : 600 heures.
— Une paire de chaussures de sport nécessite 2 heures de couture et 1 heure d'assemblage.
— Une paire de chaussures classiques nécessite 1 heure de couture et 1,5 heure d'assemblage.

Tâches :

1. Formulez le problème en programmation linéaire.
2. Résolvez le problème à l'aide de Python.

6.4 Exercice 4

Un atelier doit décider du nombre de bijoux à produire : colliers et bracelets. Chaque collier nécessite 5 grammes d'or et 2 heures de travail, rapportant un bénéfice de 200€. Chaque bracelet nécessite 3 grammes d'or et 1 heure de travail, avec un bénéfice de 120€. Les ressources disponibles sont :

— Or disponible : 300 grammes.
— Temps de travail disponible : 120 heures.

Tâches :

1. Formulez le problème en programmation linéaire.
2. Résolvez-le graphiquement.
3. Utilisez Python pour confirmer la solution.

6.5 Exercice 5

Une entreprise agricole doit décider de la quantité de blé et de maïs à planter. Le blé rapporte 500€ par hectare et le maïs 400€ par hectare. Les contraintes sont :

— Superficie totale disponible : 150 hectares.
— Budget disponible pour les semences : 20,000€.
— Coût des semences : 100€ par hectare pour le blé, 150€ par hectare pour le maïs.

Tâches :

1. Formulez le problème en programmation linéaire.
2. Trouvez la solution optimale avec Python.

7 Devoir à Rendre

7.1 Sujet du Devoir

Une entreprise de fabrication produit trois types de produits : P1, P2 et P3. Les bénéfices par unité sont respectivement de 60€, 50€ et 70€. Les produits nécessitent des temps de production sur trois machines différentes : M1, M2 et M3. Les disponibilités des machines et les temps requis par produit sont donnés dans le tableau ci-dessous :

	P1	P2	P3
Machine M1 (heures/unités)	2	1	2
Machine M2 (heures/unités)	1	2	1
Machine M3 (heures/unités)	1	1	2
Disponibilité (heures)	100	80	90

Tâches :

1. **Formulation du problème :**
   (a) Formulez le problème en termes de programmation linéaire. Définissez les variables de décision, la fonction objectif et les contraintes.
2. **Résolution avec Python :**
   (a) Écrivez un programme Python en utilisant `PuLP` pour résoudre le problème.
   (b) Commentez votre code pour expliquer chaque étape de la modélisation.
3. **Analyse et Rapport :**

(a) Rédigez un rapport détaillé présentant votre démarche, les résultats obtenus et une analyse de ces résultats.

(b) Discutez de l'utilisation optimale des machines et proposez des recommandations pour l'entreprise.

7.2 Consignes pour le Devoir

— Le programme Python doit être fonctionnel et correctement commenté.
— Le rapport doit être rédigé en LaTeX et inclure les sections suivantes :
  — Introduction.
  — Formulation mathématique du problème.
  — Présentation du code Python et explications.
  — Résultats obtenus.
  — Analyse et recommandations.
  — Conclusion.
— Le rapport doit être clair, structuré et démontrer une bonne compréhension des concepts de programmation linéaire.

8 Conclusion

Ce TP vous a permis de comprendre comment formuler et résoudre des problèmes de programmation linéaire, à la fois graphiquement et à l'aide de Python. Vous avez appris à utiliser les bibliothèques `PuLP` et `scipy.optimize` pour définir et résoudre des problèmes d'optimisation. Ces compétences sont essentielles pour optimiser des processus dans divers domaines professionnels.