Manuel de Travaux Pratiques : Résolution de Problèmes Linéaires avec IBM CPLEX

Votre Nom

23 octobre 2024

Table des matières

1 Introduction **2**

2 Objectifs du TP **2**

3 Pré-requis **2**

4 Installation de IBM CPLEX **2**
4.1 Téléchargement et Installation 2
4.2 Installation de l'API Python de CPLEX 2

5 Présentation de l'API Python de CPLEX **3**

6 Étude de Cas : Problème de Production **3**
6.1 Description du Problème 3
6.2 Formulation Mathématique 3

7 Modélisation avec l'API Python de CPLEX **3**
7.1 Étape 1 : Importation du Module 3
7.2 Étape 2 : Création du Modèle 3
7.3 Étape 3 : Définition des Variables de Décision 4
7.4 Étape 4 : Définition des Contraintes 4
7.5 Étape 5 : Résolution du Problème 4
7.6 Étape 6 : Affichage des Résultats 4
7.7 Programme Complet 5
7.8 Interprétation des Résultats 6

8 Travail à Réaliser **6**
8.1 Exercice 1 : Problème de Transport 6
8.2 Exercice 2 : Affectation de Tâches 6
8.3 Exercice 3 : Mélange de Production 6

9 Conclusion **6**

10 Annexes **7**
10.1 Ressources Utiles 7
10.2 Installation de CPLEX Studio 7

1 Introduction

La programmation linéaire est une technique d'optimisation mathématique permettant de trouver la meilleure solution possible (maximisation ou minimisation) pour un problème donné, sous certaines contraintes linéaires. IBM CPLEX est un puissant solveur d'optimisation capable de résoudre des problèmes de programmation linéaire (PL), de programmation linéaire en nombres entiers (PLNE), et d'autres types de problèmes d'optimisation.

Ce manuel de TP a pour objectif de vous guider à travers les étapes de modélisation et de résolution de problèmes linéaires à l'aide de l'outil IBM CPLEX.

2 Objectifs du TP

— Comprendre les principes de base de la programmation linéaire.
— Apprendre à modéliser un problème linéaire en utilisant le langage de modélisation de CPLEX.
— Utiliser IBM CPLEX pour résoudre des problèmes de programmation linéaire.
— Interpréter les résultats obtenus et analyser les solutions.

3 Pré-requis

— Connaissances de base en programmation linéaire.
— Notions élémentaires en algèbre linéaire.
— Familiarité avec les langages de programmation (Python sera utilisé pour l'interface avec CPLEX).
— Installation de IBM ILOG CPLEX Optimization Studio ou de l'API Python de CPLEX.

4 Installation de IBM CPLEX

4.1 Téléchargement et Installation

IBM CPLEX peut être téléchargé gratuitement pour les étudiants et les enseignants via le programme académique d'IBM. Suivez les étapes ci-dessous pour installer CPLEX :

1. Rendez-vous sur le site : `https://www.ibm.com/academic/technology`.
2. Créez un compte IBM Academic gratuit avec votre adresse e-mail universitaire.
3. Téléchargez IBM ILOG CPLEX Optimization Studio pour votre système d'exploitation.
4. Suivez les instructions d'installation fournies.

4.2 Installation de l'API Python de CPLEX

Pour utiliser CPLEX avec Python, installez le module `cplex` en utilisant la commande suivante dans votre terminal (assurez-vous que le répertoire des binaires de CPLEX est dans votre variable d'environnement PATH) :

```
pip install cplex
```

5 Présentation de l'API Python de CPLEX

L'API Python de CPLEX permet de modéliser et de résoudre des problèmes d'optimisation directement depuis Python. Elle offre une interface intuitive pour définir les variables, les contraintes et la fonction objectif.

6 Étude de Cas : Problème de Production

6.1 Description du Problème

Une entreprise fabrique deux produits, P1 et P2. Les bénéfices unitaires sont de 20€ pour P1 et de 30€ pour P2. Les produits passent par deux ateliers :

— Atelier A : P1 nécessite 2 heures, P2 nécessite 1 heure. Capacité maximale de 100 heures.

— Atelier B : P1 nécessite 1 heure, P2 nécessite 2 heures. Capacité maximale de 80 heures.

Objectif : Déterminer la quantité à produire de chaque produit pour maximiser le bénéfice total, tout en respectant les contraintes de capacité.

6.2 Formulation Mathématique

Définissons les variables de décision :

— x_1 : quantité du produit P1 à produire.

— x_2 : quantité du produit P2 à produire.

Fonction objectif à maximiser :

$$Z = 20x_1 + 30x_2$$

Sous contraintes :

$$\begin{cases} 2x_1 + x_2 \leq 100 & \text{(Capacité de l'atelier A)} \\ x_1 + 2x_2 \leq 80 & \text{(Capacité de l'atelier B)} \\ x_1 \geq 0, \quad x_2 \geq 0 & \text{(Non-négativité)} \end{cases}$$

7 Modélisation avec l'API Python de CPLEX

7.1 Étape 1 : Importation du Module

```python
import cplex
from cplex.exceptions import CplexError
```

7.2 Étape 2 : Création du Modèle

```python
def modele_production():
    try:
        # Cr ation du probl me
        problem = cplex.Cplex()

        # Sp cifier qu'il s'agit d'un probl me de maximisation
        problem.objective.set_sense(problem.objective.sense.maximize)

        #  tapes suivantes...
```

```
    except CplexError as e:
        print(e)
```

7.3 Étape 3 : Définition des Variables de Décision

```
        # Noms des variables
        noms_var = ["x1", "x2"]

        # Coefficients de la fonction objectif
        obj = [20.0, 30.0]

        # Bornes inf rieures (par d faut 0.0)
        lower_bounds = [0.0, 0.0]

        # Ajout des variables au mod le
        problem.variables.add(obj=obj, lb=lower_bounds, names=noms_var)
```

7.4 Étape 4 : Définition des Contraintes

```
        # Noms des contraintes
        noms_contraintes = ["Atelier_A", "Atelier_B"]

        # Expressions des contraintes
        lignes = [[[0, 1], [2.0, 1.0]],     # 2x1 + x2
                  [[0, 1], [1.0, 2.0]]]     # x1 + 2x2

        # C  t  s droits des contraintes
        rhs = [100.0, 80.0]

        # Sens des contraintes ('L' pour <=)
        senses = ["L", "L"]

        # Ajout des contraintes au mod le
        problem.linear_constraints.add(lin_expr=lignes, senses=senses,
                                       rhs=rhs, names=noms_contraintes)
```

7.5 Étape 5 : Résolution du Problème

```
        # R solution
        problem.solve()
```

7.6 Étape 6 : Affichage des Résultats

```
        # Affichage du statut de la solution
        print("Statut de la solution :", problem.solution.get_status())
        print(problem.solution.status[problem.solution.get_status()])

```

```
        # Affichage de la valeur optimale de la fonction objectif
        print("B n fice total optimal :", problem.solution.
            get_objective_value())

        # Affichage des valeurs optimales des variables de d cision
        valeurs_vars = problem.solution.get_values()
        for i, var_name in enumerate(noms_var):
            print(f"Quantit  optimale de {var_name} :", valeurs_vars[i
                ])
```

7.7 Programme Complet

```
import cplex
from cplex.exceptions import CplexError

def modele_production():
    try:
        problem = cplex.Cplex()
        problem.objective.set_sense(problem.objective.sense.maximize)

        # Variables de d cision
        noms_var = ["x1", "x2"]
        obj = [20.0, 30.0]
        lower_bounds = [0.0, 0.0]
        problem.variables.add(obj=obj, lb=lower_bounds, names=noms_var)

        # Contraintes
        noms_contraintes = ["Atelier_A", "Atelier_B"]
        lignes = [
            [[0, 1], [2.0, 1.0]],
            [[0, 1], [1.0, 2.0]]
        ]
        rhs = [100.0, 80.0]
        senses = ["L", "L"]
        problem.linear_constraints.add(lin_expr=lignes, senses=senses,
                                       rhs=rhs, names=noms_contraintes)

        # R solution
        problem.solve()

        # R sultats
        print("Statut de la solution :", problem.solution.get_status())
        print(problem.solution.status[problem.solution.get_status()])
        print("B n fice total optimal :", problem.solution.
            get_objective_value())
        valeurs_vars = problem.solution.get_values()
        for i, var_name in enumerate(noms_var):
            print(f"Quantit  optimale de {var_name} :", valeurs_vars[i
                ])

    except CplexError as e:
        print(e)
```

```
39
40 if __name__ == "__main__":
41     modele_production()
```

7.8 Interprétation des Résultats

Après exécution du programme, vous devriez obtenir les résultats suivants :

```
Statut de la solution : 1
optimal
Bénéfice total optimal : 2200.0
Quantité optimale de x1 : 20.0
Quantité optimale de x2 : 30.0
```

Interprétation : Pour maximiser le bénéfice total, l'entreprise doit produire 20 unités du produit P1 et 30 unités du produit P2, pour un bénéfice total de 2200€.

8 Travail à Réaliser

8.1 Exercice 1 : Problème de Transport

Une entreprise doit transporter des marchandises depuis trois usines vers quatre entrepôts. Les capacités des usines, les demandes des entrepôts, et les coûts de transport par unité sont donnés dans le tableau ci-dessous.

- Modélisez ce problème en programmation linéaire.
- Utilisez l'API Python de CPLEX pour résoudre le problème.
- Interprétez les résultats obtenus.

8.2 Exercice 2 : Affectation de Tâches

Quatre employés doivent être assignés à quatre tâches. Le coût associé à l'affectation de chaque employé à chaque tâche est donné. L'objectif est de minimiser le coût total d'affectation.

- Formulez ce problème en programmation linéaire.
- Implémentez et résolvez le problème avec CPLEX.
- Présentez la solution optimale d'affectation.

8.3 Exercice 3 : Mélange de Production

Une entreprise produit trois types d'alliages métalliques en mélangeant quatre matières premières. Chaque matière première a un coût et une disponibilité limités. Chaque alliage doit respecter une certaine composition en matières premières.

- Modélisez le problème de mélange en programmation linéaire.
- Résolvez le problème avec CPLEX.
- Analysez la solution obtenue et discutez des contraintes actives.

9 Conclusion

Ce TP vous a permis de vous familiariser avec l'utilisation de l'outil IBM CPLEX pour résoudre des problèmes de programmation linéaire. Vous avez appris à modéliser des problèmes réels, à implémenter des modèles mathématiques en Python à l'aide de l'API de CPLEX, et à interpréter les solutions obtenues.

10 Annexes

10.1 Ressources Utiles

— **Documentation de l'API Python de CPLEX** : `https://ibmdecisionoptimization.github.io/docplex-doc/`
— **Tutoriels IBM sur CPLEX** : `https://developer.ibm.com/tutorials/learn-cplex/`
— **Exemples de code** : Disponibles dans le répertoire d'installation de CPLEX sous `examples`.

10.2 Installation de CPLEX Studio

Pour une interface graphique avancée, vous pouvez utiliser IBM ILOG CPLEX Optimization Studio, qui inclut un environnement de développement intégré pour la modélisation et la résolution de problèmes d'optimisation.