Recherche Opérationnelle

TP sur l'utilisation des solveurs Google de OR-Tools dans l'environnement Python

Exercice 1

Saisissez et lancez le modèle linéaire en nombres réels suivant qui fait appel au solveur GLOP pour résoudre le problème de la cimenterie. Remarquez que les contraintes sont ajoutées en équations.

```
from ortools.linear_solver import pywraplp
def TP1_Exo1():
    # Appel du solver GLOP des problèmes linéaires en nombres réels
    solver = pywraplp.Solver('TP1_Exo1', pywraplp.Solver.GLOP_LINEAR_PROGRAMMING)
    infinity = solver.infinity()
    # Création des variables réelles x et y.
    x = solver.NumVar(0, infinity, 'x')
    y = solver.NumVar(0, infinity, 'y')
    print('Nombre des variables =', solver.NumVariables())
    # Maximize 50*x + 70*y.
    solver.Maximize(50*x + 70*y)
    # Création des contraintes.
    # 20*x + 30*y <= 360.
    solver.Add(20*x + 30*y <= 360)
    # 40*x + 35*y <= 480.
    solver.Add(40*x + 35*y <= 480)
    print('Nombre des contraintes =', solver.NumConstraints())
    solver.Solve()
    print('Solution:')
    print('Valeur optimale =', solver.Objective().Value())
    print('x =', x.solution_value())
    print('y =', y.solution_value())
TP1_Exo1()
```

Exercice 2

Saisissez et lancez le modèle linéaire en nombres entiers suivant qui fait appel au solveur CBC pour résoudre le problème de la cimenterie. Remarquez que les contraintes sont définies à travers les coefficients des variables dans les combinaisons linéaires.

```
from ortools.linear_solver import pywraplp
def TP1_Exo2():
    # Appel du solver CBC des problèmes linéaires en nombres entiers.
    solver = pywraplp.Solver('TP1_Exo2', pywraplp.Solver.CBC_MIXED_INTEGER_PROGRAMMING)
    infinity = solver.infinity()
```

```
    # Création des variables entières x et y.
    x = solver.IntVar(0.0, infinity, 'x')
    y = solver.IntVar(0.0, infinity, 'y')
    print('Number of variables =', solver.NumVariables())
    # Création de la fonction objective 50*x +70*y.
    objective = solver.Objective()
    objective.SetCoefficient(x, 50)
    objective.SetCoefficient(y, 70)
    objective.SetMaximization()
    # Création de la contrainte 20*x + 30*y <= 360.
    ct1 = solver.Constraint(-infinity, 360, 'ct1')
    ct1.SetCoefficient(x, 20)
    ct1.SetCoefficient(y, 30)
    # Création de la contrainte 40*x + 35*y <= 480.
    ct2 = solver.Constraint(-infinity, 480, 'ct2')
    ct2.SetCoefficient(x, 40)
    ct2.SetCoefficient(y, 35)
    print('Number of constraints =', solver.NumConstraints())
    solver.Solve()
    print('Solution:')
    print('Valeur optimale =', solver.Objective().Value())
    print('x =', x.solution_value())
    print('y =', y.solution_value())
    print('Advanced usage:')
    print('Problem solved in %f milliseconds' % solver.wall_time())
    print('Problem solved in %d iterations' % solver.iterations())
    print('Problem solved in %d branch-and-bound nodes' % solver.nodes())
TP1_Exo2()
```

Exercice 3

Développez et testez un modèle linéaire en nombres réels qui résout le problème du TD 4. Les contraintes doivent être ajoutées en équations.

Exercice 4

Développez et testez un modèle linéaire en nombres entiers qui résout le problème du TD 4. Les contraintes doivent être définies avec les coefficients des variables.

Exercice 5

Développez et testez un modèle linéaire en nombres réels qui résout le problème de mélange sur la fabrication de l'acier vu en cours. Ecrivez un modèle explicite en donnant séparément chaque coefficient non nul de chaque variable dans chaque contrainte. La modélisation mathématique est donnée en couleur noire dans le support du cours.

Indications :

Vous devez trouver comme solution optimale : 3524