Exemple Simple

1 Tutoriel de prise en main de PythonMIP (partie 1)

Dans ce document, vous allez voir et tester comment rentrer un programme linéaire en python avec le package PythonMIP.

Le programme linéaire en question est le suivant :

$$\begin{array}{ll} \text{Min} & -3x + y \\ \text{s.c.} & x + y = 2 \\ & 0 \leq x \leq 3 \\ & y \geq 0 \end{array}$$

1.1 Avant de rentrer votre programme linéaire

Pour travailler sur des programmes linéaires et des programmes linéaires en nombres entiers, on va utiliser la bibliothèque `mip` proposée par le paquet `pythonmip`. L'instruction suivante permet d'importer tous les outils disponibles dans cette bibliothèque.

[ ]:
```
from mip import *
```

1.2 Création et définition du modèle

Avant d'indiquer les éléments constituant le modèle (variables, objectif et contraintes), il faut créer une instance de modèle en lui précisant un nom (optionnel) et le solveur que l'on va utiliser (ici `Cbc`).

[ ]:
```
# Création du modèle vide
model = Model(name = "PLsimple", solver_name="CBC")
```

La méthode `add_var` appliquée au modèle que l'on vient de créer va permettre de lui ajouter une variable. Les paramètres de cette méthode sont les suivants :

- le paramètre `name` (optionel) permet de donner un nom à la variable ;
- le paramètre `lb` (optionel) correspond à la valeur minimale pouvant être prise par la variable. La valeur par défaut (si on ne précise pas) est 0 ;
- le paramètre `ub` (optionel) correspond à la valeur maximale pouvant être prise par la variable. La valeur par défaut (si on ne précise pas) est `INF` (aucune borne sur la valeur maximale possible équivaut à +infini) ;

- le paramètre `var_type` (optionel) correspond au type de la variable. Les valeurs possibles sont `CONTINUOUS` (variable continue),`INTEGER` (variable entière) ou `BINARY`(variable binaire). La valeur par défaut (si on ne précise pas) est `CONTINUOUS`.

Cette méthode renvoie une référence sur la variable qui a été créée, que l'on peut stocker dans une variable python. Celle-ci sera utilisée pour définir l'objectif et les contraintes.

[ ]:
```
x = model.add_var(name="x", lb=0, ub=3, var_type=CONTINUOUS)
y = model.add_var(name="y", lb=0, var_type=CONTINUOUS)
```

L'ajout d'une fonction objectif se fait en initialisant le champ `objective` du modèle avec une des fonctions `minimize` ou `maximize` prenant en paramètre l'expression linéaire définissant l'objectif à partir des références sur les variables ajoutées précédemment.

[ ]:
```
model.objective = minimize(-3*x+y)
```

Pour ajouter des contraintes au modèle, on utilise la méthode `add_constr` prenant les paramètres suivants :

- le premier paramètre est toujours l'expression définissant la contrainte ;
- le paramètre `name` (optionel) permet de donner un nom à la contrainte.

[ ]:
```
model.add_constr(x + y == 2, name="c1")
```

On peut écrire le modèle que l'on a généré dans un fichier.

[ ]:
```
model.write("exemple.lp")
```

1.3 Résolution du modèle

Il nous reste plus qu'à résoudre (optimiser) notre programme linéaire. Avec le paramètre `max_seconds`, on peut lui indiquer un temps limite en secondes. On peut également récupérer le statut de la résolution :

- `OPTIMAL` (0) ;
- `ERROR` (-1);
- `INFEASIBLE` (1);
- texttttUNBOUNDED (2).

Si on résout un PLNE des statuts supplémentaires peuvent intervenir :

- `FEASIBLE` (3) quand une solution réalisable a été trouvée, mais que l'optimalité n'a pas été prouvée;
- `INT_INFEASIBLE` (4) quand la relaxation linéaire est réalisable mais qu'il n'existe pas de solution entière;
- `NO_SOLUTION_FOUND` (5) lorsqu'aucune solution entière n'a été trouvée, mais que l'irréalisabilité n'a pas été prouvée.

[ ]:
```
satus = model.optimize(max_seconds=120)
```

Il ne reste plus qu'à récupérer et afficher la solution obtenue. Le champ `objective_value` du modèle contient la valeur de la solution trouvée. Le champ `x` de chaque variable contient la valeur de cell-ci dans la solution trouvée.

[ ]:
```
print("Valeur de la fonction objectif ", model.objective_value)
print("x = ", x.x, " y = ", y.x)
```

1.4 Bilan

Nous avons vu comment entrer et résoudre un simple programme linéaire avec `pythonmip`. Dans l'exemple suivant, nous verrons comment utiliser des tableaux pour stocker les variables et les données, et les quantificateurs pour écrire nos contraintes.