Groupe : Reynault Sies, Julien Romary

Année : 2018/2019

Projet : Optimisation

Méthode exacte pour le problème du sac à dos

Introduction

Ce document comporte les réponses aux questions du projet d'optimisation, une bibliographie/webographie ainsi qu'une explication et un diagramme de classes de l'application fournie.

Application fournie

L'application fournie est un script écrit en Python (Python 3) incluant :

- *Une implémentation de la méthode branch and bound.*
  - *Affichant l'ordre initial et la solution de la relaxation*
  - *La solution et la valeur de l'objectif*
  - *Le nombre de nœuds explorés par l'ordre de B&B*
  - *Le temps de calcul*
  - *La solution optimale*
  - *Pour chaque nœud :*
    - *La valeur de la borne supérieure*
    - *La solution relaxée*
- *La solution du problème relaxé avec les trois heuristiques.*

Indications complémentaires

Le document contient :

- *Le fichier optimisation.py contenant le script d'implémentation de Branch & Bound en python.*
- *Le fichier README.md contenant une description du projet et des indications concernant l'exécution du script.*

Table des matières

Réponses aux questions

Question n°1 :

Fonction objectif :

$$\text{Max } z = 9\,X1 + 2\,X2 + 3\,X3 + 13\,X4 + 6\,X5 + 5\,X6$$

Contraintes :

$$450\,X1 + 700\,X2 + 350\,X3 + 500\,X4 + 450\,X5 + 100\,X6 \leq 1000$$

$$X1, X2, X3, X4, X5, X6 \in \{0, 1\}$$

Avec :

- X1 qui correspond à la prise du ticket athlétisme.
- X2 qui correspond à la prise du ticket basket.
- X3 qui correspond à la prise du ticket cyclisme.
- X4 qui correspond à la prise du ticket football.
- X5 qui correspond à la prise du ticket Judo.
- X6 qui correspond à la prise du ticket natation.

Question n°2 :

Relaxation du problème sur les variables : X1, X2, …, X6 $\in$ [0, 1]

- **Première heuristique** : Ordre croissant des prix des packs

Ordre des variables :

X6 – X3 – X1 – X5 – X4 – X2

Valeurs des variables :

X6 = 1, prix = 100 et heures = 5

X3 = 1, prix = 450 et heures = 8

X1 = 1, prix = 900 et heures = 17

X5 = 0.2223, prix = 1000 et heures = 18.3

Objectif = 18.3 heures avec (1, 0, 1, 0, 0.2223, 1)

- **Deuxième heuristique** : Ordre décroissant du nombre d'heures par packs

Ordre des variables :

X4 – X1 – X5 – X6 – X3 – X2

Valeurs des variables :

X4 = 1, prix = 500 et heures = 13

X1 = 1, prix = 950 et heures = 22

X5 = 0.11111, prix = 1000 et heures = 22.6

Objectif = 22.6 heures avec (1, 0, 0, 1, 0.11111, 0)

- **Troisième heuristique** : Ordre croissant du ratio (prix/nombre d'heures)

Tableau des ratios :

Variable	X1	X2	X3	X4	X5	X6
Prix	450 €	700 €	350 €	500 €	450 €	100 €
Heures	9h	2h	3h	13h	6h	5h
Ratio	50	350	116.67	38.4	75	20

Ordre des variables :

X6 – X4 – X1 – X5 – X3 – X2

Valeurs des variables :

X6 = 1, prix = 100 et heures = 5

X4 = 1, prix = 600 et heures = 18

X1 = 0.8889, prix = 1000 et heures = 26

Objectif = 26 heures avec (0.8889, 0, 0, 1, 0, 1)

Question n°3 :

Dans le problème posé, nous cherchons à maximiser le nombre d'heures de sport auxquelles Paul peut assister tout en minimisant le coût total. Dans la troisième heuristique, on cherche dans l'ordre croissant du ratio Prix / Heures, on cherche donc d'abord :

Min (Prix / Heures)

Or, on peut remarquer que plus le prix est petit, plus le ratio sera également petit. De plus, plus le nombre d'heures est grand, plus le ratio est petit. Donc chercher Min(Prix / Heures) revient à minimiser le prix, tout en maximisant le nombre d'heures.

C'est pourquoi la troisième heuristique permet de récupérer un résultat optimal pour le problème relaxé.

Les deux autres heuristiques ne sont pas optimales :

- **Première heuristique** : Ordre croissant des prix des packs

Contre-exemple :

Soit les données suivantes :

- Athlétisme → 500€ pour 1h (X1)
- Basket → 500€ pour 1h (X2)
- Cyclisme → 1000€ pour 5000h (X3)
- Football → 500€ pour 1h (X4)
- Judo → 500€ pour 1h (X5)
- Natation → 500€ pour 1h (X6)

Avec cette heuristique, cela nous donne :

X1 = 1, avec 500€ pour 1h

X2 = 1, avec 500€ pour 1h

Puis X3, X4, X5, X6 à 0.

L'objectif serait alors de 2 heures.

Supposons que la première heuristique soit la plus optimale.

Il y a alors une contradiction, en effet, si X3 était égal à 1, le nombre d'heures serait à 5000 heures ce qui est plus grand que 2h.

Donc la première heuristique n'est pas optimale.

- **Deuxième heuristique** : Ordre décroissant du nombre d'heures par packs

Contre-exemple :

Soit les données suivantes :

- Athlétisme → 200€ pour 10h (X1)
- Basket → 200€ pour 10h (X2)
- Cyclisme → 1000€ pour 20h (X3)
- Football → 200€ pour 10h (X4)
- Judo → 200€ pour 10h (X5)
- Natation → 200€ pour 10h (X6)

Avec cette heuristique, cela nous donne :

X3 = 1, avec 20h et 1000€

Puis X1, X2, X4, X5, X6 à 0.

L'objectif serait alors de 20 heures.

Supposons que la deuxième heuristique soit la plus optimale.

Il y a alors une contradiction, en effet, on remarque qu'on peut prendre X1, X2, X4, X5, X6 à 1 et X3 à 0. On aurait alors 10 + 10 +10 + 10 +10 = 50 heures, ce qui est plus grand que 20 heures.

Donc la deuxième heuristique n'est pas optimale.

Question n°4 :

- Solution du problème relaxé avec chaque Heuristique :

```
Heuristique 1

Ordre : x6 x3 x1 x5 x4 x2
Valeurs :

x1 = 1
x2 = 0
x3 = 1
x4 = 0
x5 = 0.2222222222222222
x6 = 1

Duree de la solution = 18.333333333333336
Cout de la solution = 1000.0

Heuristique 2

Ordre : x4 x1 x5 x6 x3 x2
Valeurs :

x1 = 1
x2 = 0
x3 = 0
x4 = 1
x5 = 0.1111111111111111
x6 = 0

Duree de la solution = 22.666666666666668
Cout de la solution = 1000.0

Heuristique 3

Ordre : x6 x4 x1 x5 x3 x2
Valeurs :

x1 = 0.8888888888888888
x2 = 0
x3 = 0
x4 = 1
x5 = 0
x6 = 1

Duree de la solution = 26.0
Cout de la solution = 1000.0
```

- Solution finale trouvée par le programme :

```
Fin du branch and bound
Nombre de noeuds : 17
Temps de calcul : 144.0 ms
Solution optimale :

Valeurs :

x1 = 1
x4 = 1
x6 = 0
x5 = 0
x3 = 0
x2 = 0

Duree de la solution = 22
Cout de la solution = 0
```

L'application affiche les informations nœud par nœud, il faut l'exécuter avec une version de Python supérieure ou égale à 3.

Question n°5 :

On peut modéliser cette nouvelle contrainte avec une nouvelle variable, comme ceci :

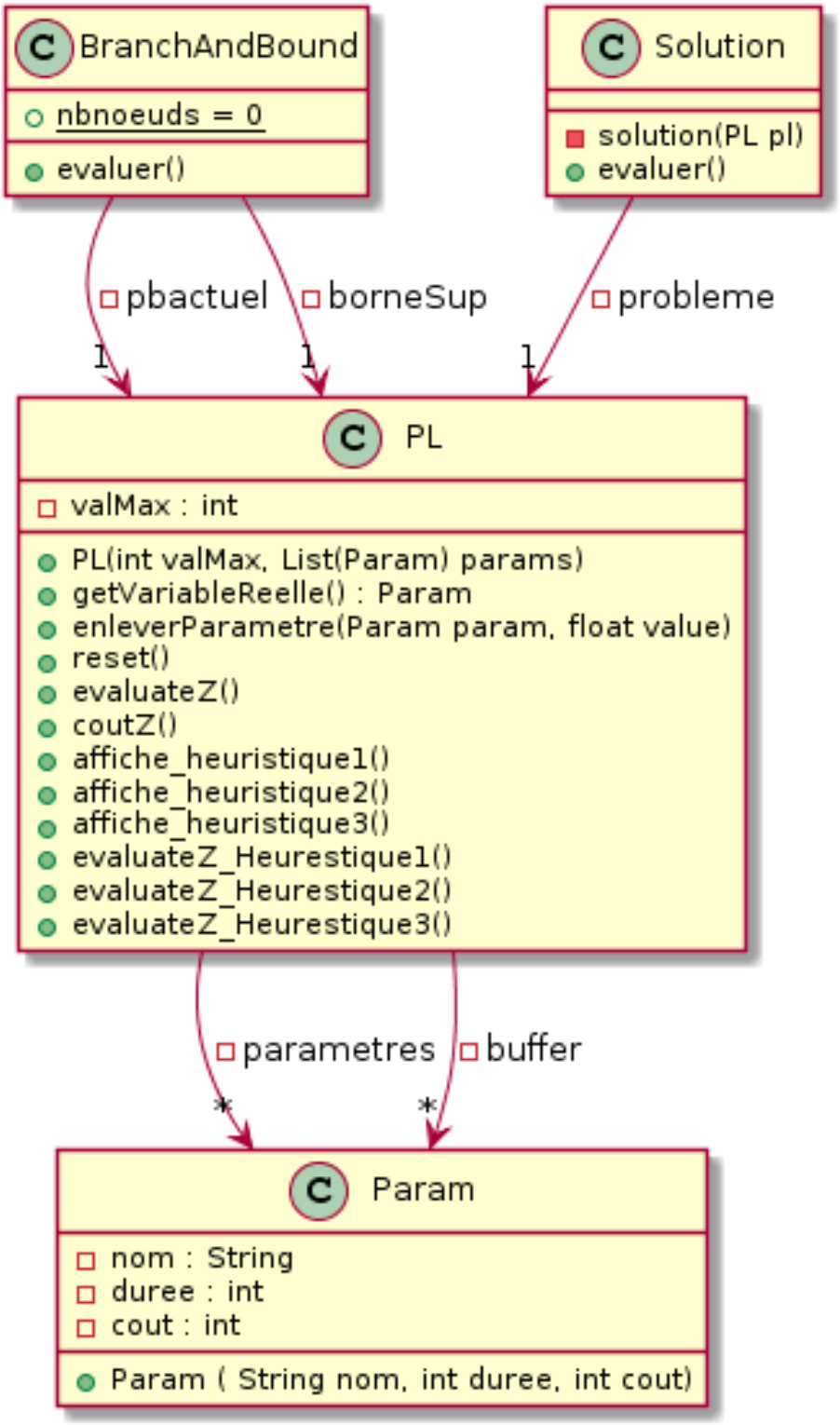
Athlétisme → 9h, 450€	(X1)
Basket → 2h, 700€	(X2)
Cyclisme → 3h, 350€	(X3)
Football → 13h, 500€	(X4)
Judo → 6h, 450€	(X5)
Natation → 5h, 100€	(X6)
Natation-Athlétisme → 11h, 550€	(X7)

On ajoute alors une contrainte qui prend en compte X7 :

$9X1 + 5X6 + 11X7 \leq 11$

Cette contrainte permet d'obliger X1 et X6 à être à 0 lorsque X7 est à 1 et inversement dans un problème non relaxé.

Diagramme de classes



Annexe

BranchAndBound : Classe représentant la résolution d'un problème avec Branch & Bound.

Attributs

Nbnoeud : Nombre de nœuds de l'arbre.

Pbactuel : Problème le plus optimal trouvé à l'instant t.

BorneSup : Borne supérieure correspondant au problème relaxé avec l'heuristique numéro 3.

Méthode

Évaluer () : Méthode permettant d'évaluer le problème fourni. (Initialisation de l'arbre)

Solution : Solution représentant un nœud dans l'arbre du Branch & Bound.

Attributs

Problème : Problème en cours d'évaluation.

Méthode

Évaluer () : Méthode permettant d'évaluer le problème du nœud en cours.

PL : Classe représentant un problème linéaire.

Attributs

ValMax : Valeur maximale correspondant à la contrainte d'argent posée par le problème.

Paramètres : Paramètres du problème linéaire, un paramètre est un nom de variable, un coût et une durée, celui-ci est lié par un dictionnaire à sa valeur. (Variable → Valeur entre 0 et 1)

Buffer : Dictionnaire paramètre et valeur ne contenant que les variables déjà fixées par l'algorithme de Branch & Bound.

Méthodes

GetVariableReelle () : Méthode qui permet de récupérer la valeur réelle du problème relaxé.

EnleverParamètre (Param param, float value) : Méthode permettant d'enlever un paramètre de la liste des paramètres et de le mettre dans buffer en spécifiant la valeur fixe. (0 ou 1)

Reset () : Méthode permettant de remettre à 0 toutes les variables du problème.

EvaluateZ () : Méthode permettant de calculer et de retourner le nombre d'heures total.

CoutZ () : Méthode permettant de calculer le coût total.

Affiche_heuristique1 () : Méthode permettant d'afficher l'ordre des variables pour la première heuristique.

Affiche_heuristique2 () : Méthode permettant d'afficher l'ordre des variables pour la deuxième heuristique.

Affiche_heuristique3 () : Méthode permettant d'afficher l'ordre des variables pour la troisième heuristique.

EvaluateZ_Heurestique1 () : Méthode permettant de calculer les valeurs de chaque variable contenue dans la liste des paramètres avec la première heuristique.

EvaluateZ_Heurestique2 () : Méthode permettant de calculer les valeurs de chaque variable contenue dans la liste des paramètres avec la deuxième heuristique.

EvaluateZ_Heurestique3 () : Méthode permettant de calculer les valeurs de chaque variable contenue dans la liste des paramètres avec la troisième heuristique.

Param : Classe représentant un paramètre du problème linéaire, soit un nom de variable et le coût et la durée associés.

Attributs

Nom : Nom du paramètre. (x1, x2, x3 ….)

Coût : Coût lié à la variable.

Durée : Durée liée à la variable.

Bibliographie / Webographie

- Documentation en ligne du langage Python :

  https://www.python.org/doc/

Consulté le 07/04/2019

- Tutoriels sur le site tutorials point : (tuto pour python et l'utilisation des dictionnaires notamment)

  https://www.tutorialspoint.com/python/

Consulté le 07/04/2019