Université d'Evry Val d'Essonne
L1 SPI - Année universitaire 2018/2019

Modélisation et Simulation - EC222

TP2 - Optimisation continue et combinatoire

Consignes :

- pour toute question ou remarque : gaetan.hello@univ-evry.fr

1 Conception d'un circuit d'adduction d'eau

On considère deux villes, V_1 et V_2, situées respectivement à des distances $d_1 = 5\,\text{km}$ et $d_2 = 2\,d_1$ d'un même fleuve. Celui-ci s'écoule de manière rectiligne et les deux villes sont situées sur la même rive. La distance entre les points du fleuve les plus proches de chacune des deux villes est $L = 3\,d_1$.

Les deux villes souhaitent construire un système d'adduction d'eau permettant d'acheminer l'eau du fleuve vers chacune d'entre elles. Le système est constitué d'une pompe placée sur le fleuve et de deux canalisations rectilignes reliant la pompe aux deux villes.

Votre objectif consiste à définir la position de la pompe qui minimisera la longueur totale de canalisation.

1.1

Traduire l'énoncé en langage naturel vers un modèle géométrique.

1.2

En paramétrant la position de la pompe au moyen d'une quantité x, exprimer alors la fonction-objectif du problème d'optimisation en fonction de $d1$, $d2$, L et x.

1.3

Tracer au moyen de Python la courbe donnant la longueur totale de canalisation en fonction de la position de la pompe sur le fleuve. Commenter. Quelle position approximative suggéreriez-vous de choisir et pourquoi ?

1.4

Exprimer formellement le problème sous forme d'un problème d'optimisation continue où la position optimale sera liée à la dérivée de la fonction-objectif. Tracer alors la dérivée de la fonction-objectif. Commenter.

1.5

Au moyen de l'algorithme de la dichotomie, proposer une approximation numérique de la solution optimale précise à $\varepsilon \approx 10^{-6}$.

1.6

Au moyen de l'algorithme de Newton-Raphson, proposer une approximation numérique de la solution optimale précise à $\varepsilon \approx 10^{-6}$. Pour ceux en difficulté avec le calcul de la dérivée seconde, vous pouvez la calculer formellement sur le site https://www.wolframalpha.com/ avec l'expression `D[sqrt[d1^2+x^2]+sqrt[d2^2+(L-x)^2],{x,2}]`.

1.7

Commenter et comparer le comportement des 2 méthodes.

2 Problème d'optimisation combinatoire du "knapsack"

Au même titre que le problème du voyageur de commerce ("travelling salesman problem"), le problème du sac-à-dos ("knapsack problem") a fait l'objet de nombreuses recherches en raison de son large spectre d'applications tant théoriques que pratiques. Il consiste à maximiser la valeur du contenu d'un sac en le remplissant par des objets issus d'une liste d'éléments de masse et valeur données tout en respectant la masse limite que peut supporter le sac. On notera ainsi :

- m_{max} la masse maximale que le sac peut supporter,
- n le nombre d'objets de la collection,
- m_i et v_i respectivement la masse et la valeur de l'objet d'indice i,
- $c_i \in \{0, 1\}$ le coefficient indiquant si l'objet i est inclus (1) ou non (0) dans le sac,
- $M = \{m_i\}_{1 \leq i \leq n}$ la liste des masses pour les différents objets,
- $V = \{v_i\}_{1 \leq i \leq n}$ la liste des valeurs pour les différents objets,
- $C = \{c_i\}_{1 \leq i \leq n}$ la liste des coefficients pour les différents objets (0 ou 1).

La fonction que l'on souhaite maximiser est alors la suivante (valeur cumulée des objets présents dans le sac) :

$$v(C) = \sum_{i=1}^{n} c_i \, . v_i \tag{1}$$

Le choix fait pour C doit respecter la contrainte d'admissibilité du chargement :

$$\sum_{i=1}^{n} c_i \, . m_i \leq m_{max} \tag{2}$$

On se propose ici de résoudre le problème avec :

$$\begin{aligned} n &= 6 \\ m_{max} &= 10 \\ M &= \{6, 2, 3, 5, 4, 1\} \\ V &= \{10, 11, 12, 13, 14, 15\} \end{aligned} \tag{3}$$

Votre travail consiste à rédiger un script Python permettant de trouver la valeur de C maximisant (1) en respectant (2) pour les paramètres (3). La nature de l'algorithme est laissée à votre discrétion sachant qu'un algorithme de parcours exhaustif des combinaisons s'avère ici relativement simple à implémenter et peu coûteux en temps d'exécution...

2.1

Trouver la solution sur le papier par une méthode de parcours exhaustif de l'ensemble des solutions. (pour simplifier le travail bien analyser au préalable les couples de propriétés (m_i, v_i) pour chaque objet).

2.2

Formaliser sur le papier l'algorithme de la méthode.

2.3

Implémenter l'algorithme en langage Python.