# Programmation Linéaire

Il nous a été demandé, dans le contexte d'un projet de programmation linéaire, de minimiser la distance de manhattan entre un nombre donné de points relais et de potentiels clients voulant utiliser ces relais de la façon la plus optimisée possible.

Dans un premier temps, nous avons décidé de nous focaliser sur la résolution du problème n'utilisant qu'un seul point relais à positionner pour deux clients. Trouver une approximation de la solution pour une paire de clients nous permettra ensuite d'envisager une solution plus poussée permettant de résoudre le problème pour $n$ clients.

Pour faciliter notre apprentissage de la programmation linéaire et afin de mieux comprendre et visualiser son utilisation, nous avons utilisé un solver en ligne permettant la résolution de programmes linéaires.
C'est ainsi que nous avons implémenté pour la première fois une version minimisant la distance entre deux paires de clients et un point relais.

Une fois la simulation opérationnelle, nous avons codé une version gardant la même logique en Python.
La différence majeure fût de travailler avec des matrices de coefficients liés implicitement à des variables au lieu de travailler directement avec des variables déclarées en dur dans le programme. C'est ensuite grâce à Scipy et $linprog()$ que nous avons pu résoudre le programme linéaire via la méthode simplex.

**Démonstration**

Soit

- $n$ le nombre de clients.
- $m$ le nombre de points relais.
- $vn$ le nombre de variables associé à $n$.
- $vm$ le nombre de variables associé à $m$.
- $c$ le nombre de contraintes associé à $n$.
- $d$ la dimension des points (2D) = 2.

Posons les matrices de dimensions (col, row)

- $C \ : (n * vn \ + \ m * vm)$
- $A_{ub} : (n * vn \ + \ m * vm, \ c * (n * vn \ + \ m * vm))$
- $b_{ub} : (n * vn \ + \ m * vm)$
- $A_{eq} : (n * vn \ + \ m * vm, \ n * d)$
- $b_{eq} : (n * d)$

$C$ représente l'équation à minimiser.
$A_{ub}$ représente la partie gauche des inéquations de contraintes.
$b_{ub}$ représente la partie droite des inéquations de contraintes.

$A_{eq}$ représente la partie gauche des équations de contraintes.
$b_{eq}$ représente la partie droite des équations de contraintes.

Nous cherchons à minimiser l'équation: $C = d1 + d2$
Le programme linéaire est ensuite résolu en utilisant

$$linprog(C,\ A\_ub = A\_ub,\ b\_ub = b\_ub,\ A\_eq = A\_eq,\ b\_eq = b\_eq,\ bounds = bounds)$$

**Exemple**

Posons deux points clients:

$A = (10,\ 15)$

$B = (20,\ 30)$

Nous obtenons:

$x = 14,2$

$y = 20,2$ ↠ Ce point se trouve bien parmi les points optimaux se situant entre $A$ et $B$

.

# Ressources

Online solver :
https://online-optimizer.appspot.com/

Scipy::linprog :
https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.linprog.html?fbclid=IwAR36wehNgbORocxMjD61liqV62YU2d_3UIv6Wf_YOZsBL-pBDRXk2NQPETc

LP with absolute values:
https://optimization.mccormick.northwestern.edu/index.php/Optimization_with_absolute_values#:~:text=Absolute%20values%20in%20constraints

# Authors

Bilot Tristan
Delfau Nora
Faivre Romaric
Salemi Enzar
Thambithurai Madushan