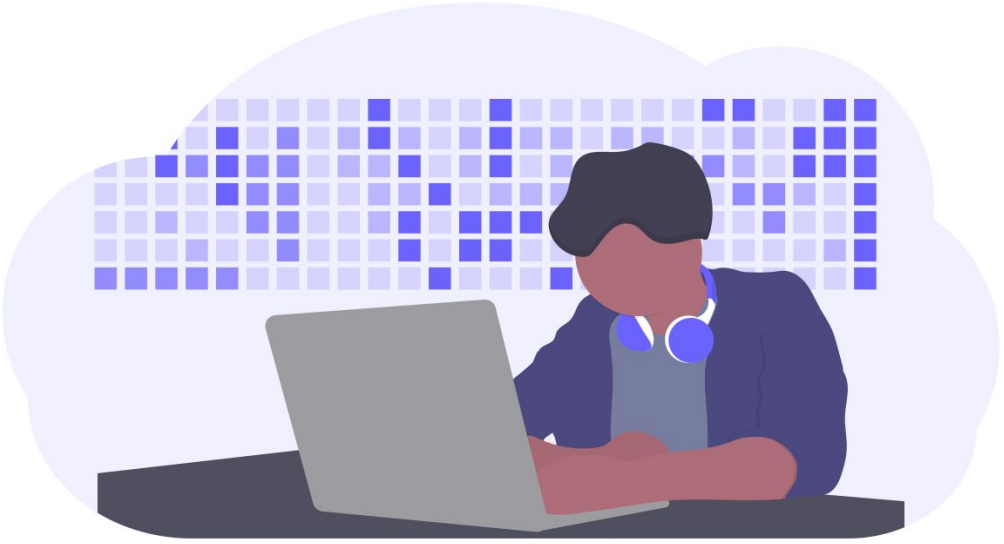


PROJET RO

Jean Lagniaux et Theo Denes Grp Alternance 2C

Sommaire :

Questions du sujet:

A quel problème théorique est associé celui proposé par l'entreprise ?

- Le problème théorique proposé par l'entreprise est un problème de maximisation. On peut donc associer ça à un primal.

Quelle structure de données allez-vous utiliser pour représenter les informations des scénarios :

- On a organisé les informations des fichiers data sous forme de graphe.

De quoi doit-on s'assurer pour les clients ? Pour les dépôts ?

- Pour les clients et pour les dépôts, on doit s'assurer que leur demande ou leur stock ne passe pas en dessous ou au dessus de 0 (rupture de stock ou servir plus que nécessaire)

De quelle manière allez-vous représenter le chemin de votre camion ? La quantité qu'il transporte sur les routes pour lesquelles il est passé ?

- Nous avons voulu créer un dictionnaire avec pour clé les coordonnées de départ et d'arrivée (la route), et comme valeur, la quantité transportée sur cette même route.

Programme linéaire :

Variables :

La route :

- La route est une variable binaire. En effet, une route peut être empruntée une seule fois ou ne pas être empruntée du tout. Ainsi, nous devons récupérer cette information pour déterminer par la suite le coup de transport total.

  R(i,j) = 1 si route empruntée, 0 sinon

Les dépôts :

- Un dépôt est également une variable binaire. En effet, il est possible qu'un dépôt soit inaccessible suivant le système dans lequel on se trouve. Nous avons donc choisi de modéliser cela en précisant si oui ou non, ce dépôt sera utilisé.

  Di = 1 si dépôt en fonctionnement, 0 sinon

Les clients :

- Un client peut également être inaccessible suivant le système (data) dans lequel on doit résoudre le problème. Dans la même logique que les dépôts précédemment évoqués, on va choisir de symboliser les clients comme une variable binaire. Y avons-nous accès ?

Ci = 1 si le client doit être livré, 0 sinon

Le stock du camion sur une route :

- Nous avons besoin de récupérer le stock du camion sur une route en question. En effet, celui-ci est variable puisque le camion peut passer entre chaque route par des dépôts ou des clients. Ainsi, le stock va varier.

StC(i,j) = la quantité de GPU qui circule entre un point i et un point j

Contraintes :

Le stock du camion sur une route i doit être inférieur ou égal à la capacité de cette même route i :

StC(i,j) <= Capacité du camion (constante)

Le stock du camion sur une route est toujours inférieur ou égal à la capacité de stockage du camion.

StC(i,j) <= Capacité de la route (constante)

S'assurer qu'une route utilisée 1 fois ne sera pas empruntable par la suite. Ainsi, la route étant une variable binaire, on veut s'assurer que la somme des éléments de la liste des routes utilisée est inférieure à 1.

∑R(i,j) <=1

(Supposons que R(i,j) ait plusieurs valeurs binaires affectées, on va sommer R(i,j) pour chaque (i,j) et vérifier qu'il est inférieur ou égal à 1).

Mise à jour des stocks du dépôt : on sait que le stock d'un dépôt est exprimé par une valeur négative. On va donc s'assurer que la valeur finale du stock de chaque dépôt n'est pas supérieur à 0. 0 étant le stock vide d'un dépôt.

Di * Stock du dépôt <= 0

Mise à jour de la demande client : la demande client est en revanche une valeur positive. On va donc s'assurer que la valeur de cette demande est toujours supérieure ou égale à 0.

Besoin client >= 0

S'assurer que le client est entièrement servi.

Ci * besoin client = 0

Objectif:

L'objectif est de maximiser les bénéfices. Ainsi, l'objectif est de maximiser les ventes et de minimiser le coût de transport.

Pour ce faire, nous avons choisi de maximiser le résultat des ventes moins les coûts de transport.

Voici donc la formule : que l'on doit maximiser

Formule du calcul du chiffre d'affaire :

- **1000 * Ci * Besoins client i** (1000 * Besoins des clients servis)

Formule du calcul des coûts :

- **R(i,j) * Coût de l'essence sur la route i + R(i,j) * taxe douanière sur la route i * StC(i,j)**

Formule du calcul du bénéfice (fonction objective): Chiffre d'affaire - coûts

- **$\sum$[1000 * Ci * Besoin client i] - [R(i,j) * Coût de l'essence sur la route i + R(i,j) * taxe douanière sur la route i * StC(i,j)]**

Structure du programme :

Fichier Reader.py :

Fichier qui permet d'extraire à partir des fichiers de données un graphe que nous allons utiliser pour résoudre le problème linéaire. Le graphe va être modélisé grâce à la bibliothèque Networkx. De plus, nous avons donné la possibilité à l'utilisateur de consulter le graphe avec la librairie matplotlib ou encore grâce à un export au format IML qui peut être lu sous cytoscape.

Fichier truck_pulp.py :

Dans ce fichier, nous avons la définition du programme linéaire ainsi que le solveur qui se base sur la librairie Pulp. Nous utilisons donc les données que nous avons exportées dans le fichier Reader.py afin de définir nos variables et contraintes. Quant à lui, le solveur va résoudre le problème linéaire et afficher les données que nous voulons partager avec les utilisateurs (la fonction objective, un dictionnaire des routes avec la quantité de GPU sur ses routes…)

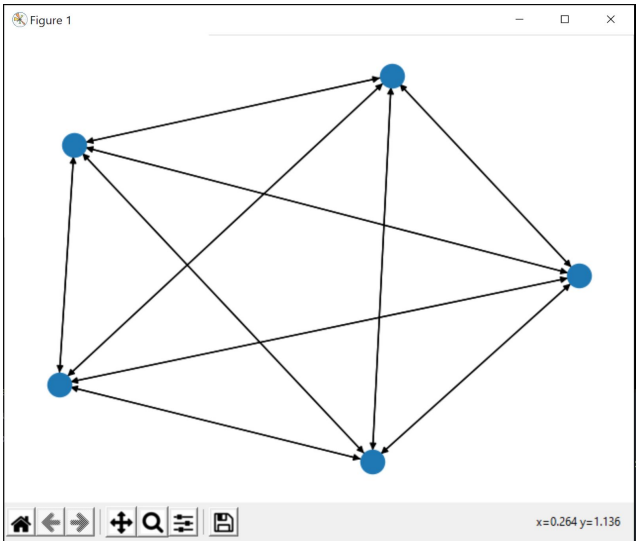
Fichier test.py :

Ce fichier nous sert de fichier d'entrée pour faire fonctionner notre projet. C'est dans ce fichier que l'on va pouvoir lancer notre solveur et résoudre le problème avec les différentes set de données mises à notre disposition.

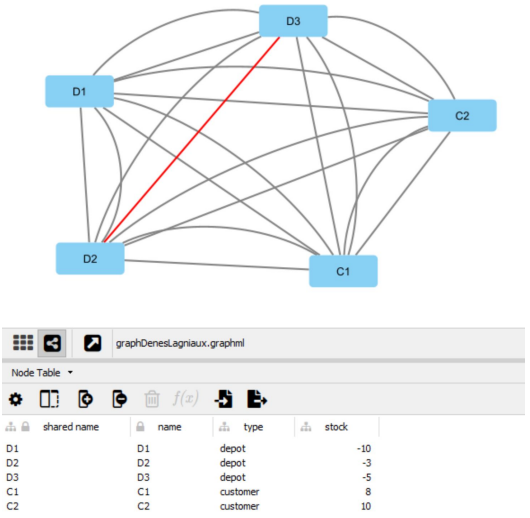
Présentation et analyse des résultats :

Nous n'avons malheureusement pas réussi à obtenir de résultats lors de l'exécution de ce programme. En effet, nous avons des erreurs que nous ne réussissons pas à résoudre. Malgré tout, il nous semblait que le choix des variables, des contraintes et de la fonction objectif aurait pu mener à des résultats pertinents.

Pour ce qui de l'exportation des données, nous avons eu des résultat très concluants :



Le graphe sous matPlotLib



Le graphe sous Cytoscape

Dysfonctionnements et hypothèses:

Nous n'avons donc pas réussi à faire fonctionner notre programme :

```
Traceback (most recent call last):
  File "C:\EspacePerso\CODE\MIAGE\RO\RO-2020\M1_MIAGE_2020_RO_Projet\test.py", line 13, in <module>
    obj, dicts_var, truck_stock_onRoad  = solv.solve_truck_problem(instance_path)
  File "C:\EspacePerso\CODE\MIAGE\RO\RO-2020\M1_MIAGE_2020_RO_Projet\projet_RO_LAGNIAUX_JEAN_DENES_THEO\truck_pulp.py", line 109, in
 solve_truck_problem
    prob, truck_stock_onRoad = def_truck_problem(graph, entete)
  File "C:\EspacePerso\CODE\MIAGE\RO\RO-2020\M1_MIAGE_2020_RO_Projet\projet_RO_LAGNIAUX_JEAN_DENES_THEO\truck_pulp.py", line 63, in
def_truck_problem
    prob += pl.lpSum(1000 * use_customer[c] * customer_need[c] for c in list_customer) - pl.lpSum((use_road[(i, j)] * dicts_route[(i
,j)]['gas'] + (use_road[(i, j)] * dicts_route[(i,j)]['tax'] * truck_stock_onRoad[(i,j)]) for (i,j) in list_route))
  File "C:\Users\cobi-\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.9_qbz5n2kfra8p0\LocalCache\local-packages\Python39\s
ite-packages\pulp\pulp.py", line 2075, in lpSum
    return LpAffineExpression().addInPlace(vector)
  File "C:\Users\cobi-\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.9_qbz5n2kfra8p0\LocalCache\local-packages\Python39\s
ite-packages\pulp\pulp.py", line 765, in addInPlace
    for e in other:
  File "C:\EspacePerso\CODE\MIAGE\RO\RO-2020\M1_MIAGE_2020_RO_Projet\projet_RO_LAGNIAUX_JEAN_DENES_THEO\truck_pulp.py", line 63, in
<genexpr>
    prob += pl.lpSum(1000 * use_customer[c] * customer_need[c] for c in list_customer) - pl.lpSum((use_road[(i, j)] * dicts_route[(i
,j)]['gas'] + (use_road[(i, j)] * dicts_route[(i,j)]['tax'] * truck_stock_onRoad[(i,j)]) for (i,j) in list_route))
  File "C:\Users\cobi-\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.9_qbz5n2kfra8p0\LocalCache\local-packages\Python39\s
ite-packages\pulp\pulp.py", line 838, in __mul__
    return self * LpAffineExpression(other)
  File "C:\Users\cobi-\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.9_qbz5n2kfra8p0\LocalCache\local-packages\Python39\s
ite-packages\pulp\pulp.py", line 826, in __mul__
    raise TypeError("Non-constant expressions cannot be multiplied")
TypeError: Non-constant expressions cannot be multiplied
```

Comme on peut le voir, nous avons plusieurs erreurs dans notre programme.

Nous ne comprenons pas, par exemple, pourquoi le programme nous dit qu'il est interdit de multiplier deux expressions non constantes et nous n'avons pas réussi à les identifier. De

plus, avec notre fonction objectif, nous utilisons beaucoup de maps et de listes ; malgré nos différents tests, nous ne sommes pas certains de sortir des expressions calculables. C'est sûrement cette utilisation trop forte de dictionnaires dans nos variables qui rend notre code compliqué.

Nous émettons aussi un hypothèse de non fonctionnement sur notre variable "truck_stock_onRoad". Certes, elle prend comme clé chaque route donnée, mais nous ne savons pas quand est ce que le nombre de GPU transporté sur cette route sera affecté en valeurs. Mais nous avons quand même émis des contraintes pour cette variable. Le reste de notre programme semble cohérent et nous pensons que nos variables et contraintes le sont aussi mais nous n'avons pas réussi à nous débloquer.

Nous avons essayé le plus possible d'avancer dans le projet en réalisant un code qui nous semblait le plus fonctionnel possible. Cependant, les résultats obtenus ne sont pas à la hauteur de notre implication. Ayant une erreur dans la fonction objectif, nous ne pouvons évidemment pas dégager de résultats. Pourtant, nous savons que cette fonction peut répondre à ce problème.