Rapport RO TP1

Olivier Louis-Clément, Guttierez Tom

Decembre 2024

Contents

1 Introduction

Lors de ce TP nous avons pris les problèmes modélisés en TD et nous les avons résolu grace au solveur GLPK. Nous présentons notre démarche de modélisation et nos résultats ci dessous.

2 Problème d'assemblage

Pour le premier problème nous devons maximiser le profit d'une usine de fabrication de vélo, en trouvant le nombre de vélo cargo et de vélo normaux à produire sur une semaine.

on a deux variables :
- $x1$: le nombre de vélo cargo
- $x2$: le nombre de vélo normaux

La fonction a maximiser est la suivante : $700 \times x2 + 300 \times x1$

avec les contraintes suivantes :

$$\begin{cases} 0.06 \times x1 + 0.05 \times x2 & \leq 60 \\ 2.5 \times x1 + x2 & \leq 1500 \\ x1 & \leq 700 \end{cases}$$

on a alors le fichier *PbVelo.lp.txt* suivant :

```
\* Problem: Fabrication de Velo *\

Maximize
Benefice: 700 Q1 + 300 Q2

Subject To
HeureTravail : 0.06 Q1 + 0.05 Q2 <= 60
PlaceRangement : 2.5 Q1 + Q2 <= 1500
NbCargo : Q1 <= 700

End
```

Figure 1: Code du fichier PbVelo.lp.txt

On utilise un fichier ".lp" car on ne veut résoudre le problème qu'une seule fois. En supposant que les coûts ne change pas d'un jour a l'autre, la solution optimale reste la même.

Voici un tableau avec les résultats pour un Problème Linéaire, ou un Problème Linéaire en nombre entiers :

	PL	PLNE
Vélo Cargo	923.077	920
Vélo Normaux	230.769	232

3 Problème de distribution de taches

Dans ce problème nous devons maximiser le bonheur de nos N employés. Pour cela ils ont noté N tâches de 1 à 10 en fonction de leur envie de la réaliser.

On a alors comme variable une matrice X de taille $N \times N$ remplie de 0 et de 1. nous devons alors maximiser la fonction suivante :

$$\sum_{i,j=1}^{N} X_{ij} \times A_{ij} \qquad \text{où } A_{ij} \text{ est la note qu'a mit l'employé i a la tâche j}$$

la contrainte a respecter est alors que X ne peut avoir qu'une valeur non nulle par ligne et colonne (un employé ne doit travailler que sur une tache et une tâche ne peut avoir qu'un employé qui travaille dessus). On peut représenter ces contraintes de la manière suivante :

- $\forall i, j \in [1, N],\ X_{ij} \in \{0, 1\}$
- $\forall j \in [1, N], \qquad \sum_{i=1}^{N} X_{ij} = 1$
- $\forall i \in [1, N], \qquad \sum_{j=1}^{N} X_{ij} = 1$

On peut alors créer les fichier *PbTaches.mod.txt* ainsi que le jeu de donnée de test.

```
data;

set PERSONNES :=
1
2
3
4
5;

set TACHES :=
T1
T2
T3
T4
T5;

param mattaches: T1 T2 T3 T4 T5 :=
1 8 7 3 5 6
2 4 6 2 10 4
3 10 4 8 10 3
4 7 5 4 1 2
5 1 3 5 7 8;

end;
```

Figure 2: Code du fichier DataTaches.dat.txt

```
################################  Sets  ################################

set TACHES;

set PERSONNES;

################### Variables ###################

var Q{i in PERSONNES, j in TACHES}, binary;


###################  Constants: Data to load   ########################

param mattaches{i in PERSONNES, j in TACHES};

################### Constraints ###################


s.t. RespecteConditionsColonnes{j in TACHES}:
  sum{i in PERSONNES} Q[i,j] = 1;

s.t. RespecteConditionsLignes{i in PERSONNES}:
  sum{j in TACHES} Q[i,j] = 1;


###### Objective ######

maximize Bonheur:
                sum{i in PERSONNES} sum{j in TACHES} Q[i,j]* mattaches[i,j];


end;
```

Figure 3: Code du fichier PbTaches.mod.txt

On utilise pour la résolution de ce problème un fichier ".mod" qui permet d'avoir un jeu de donnée détaché car on veut pouvoir résoudre ce même problème sur plusieurs jeu de donnée. Par exemple, si il y a des nouvelles taches a réaliser toute les semaines.

On obtient alors le résultat suivant pour le jeu de donnée ci dessus :

	$T1$	$T2$	$T3$	$T4$	$T5$
1	0	1	0	0	0
2	0	0	0	1	0
3	0	0	1	0	0
4	1	0	0	0	0
5	0	0	0	0	1

4 Distribution de fluides

4.1 Cas Particulier 1

Pour le problème suivant nous voulons distribuer différents fluides (F_i) a différents clients (D_i). Pour cela nous devons les acheter a différents magasins (M_i) où les prix des fluides sont différents. Nous cherchons alors a faire payer un coût minimal au clients. Voici le jeu de données en exemple :

	F1	F2
D1	2	0
D2	1	3

(a) Fluides demandés par commande

	F1	F2
M1	2.5	1
M2	1	2
M3	2	1

(b) Stocks de fluides par magasin

	F1	F2
M1	1	1
M2	2	3
M3	3	2

(c) Coûts unitaires par magasin d'origine

Figure 4: Données exemple pour le problème

On a besoin de définir les variables et constantes suivantes :
- M_N, F_N, D_N respectivement le nombre de magasins, de fluides et de demandeurs
- X_{ij} la quantité de fluide j acheté au magasin i
- p_{ij} le prix a l'unité du fluide j au magasin i
- M_{ij} la quantité de fluide j au magasin i

On cherche alors a minimiser : $\sum_{i=1}^{M_N} \sum_{j=1}^{F_N} X_{ij} \times p_{ij}$

Il y a alors deux contraintes a modéliser. D'abord, on ne peut pas acheter plus de Fluides à un magasin qu'il en a en stock. Ensuite il faut que la quantité de chaque fluide acheté correspondent a la quantité commandée. On obtient alors :

- $\forall (i,j) \in [1, M_N] \times [1, F_N]$, $X_{ij} \leq M_{ij}$

-$\forall j \in [1, F_N]$, $\sum_{i=1}^{M_N} X_{ij} = \sum_{i=1}^{D_N} D_{ij}$

On peut alors écrire le fichier *PbFluides.mod.txt*. On utilise ici aussi un fichier ".mod" car on veut pouvoir résoudre le problème plusieurs fois avec des jeux de données différents (chaque nouvelle commande a un nombre de demandeur différent, voulant des quantités différentes de fluides).

```
4   ############################  Sets  ############################
5
6   set FLUIDES;
7
8   set DEMANDEURS;
9
10  set MAGASINS;
11
12  ################## Variables ##################
13
14  var Q{i in MAGASINS, j in FLUIDES}, >=0;
15
16
17  ##################  Constants: Data to load   ######################
18
19  param fldem{i in DEMANDEURS, j in FLUIDES};
20
21  param flmagst{i in MAGASINS, j in FLUIDES};
22
23  param flmagp{i in MAGASINS, j in FLUIDES};
24
25  ################## Constraints ##################
26
27
28  s.t. quantitedem{j in FLUIDES}:
29    sum{i in MAGASINS} Q[i,j] = sum{i in DEMANDEURS} fldem[i,j];
30
31  s.t. quantitemag{i in MAGASINS,j in FLUIDES}:
32    Q[i,j] <= flmagst[i,j];
33
34
35  ###### Objective ######
36
37  minimize PrixFluide:
38                  sum{i in MAGASINS} sum{j in FLUIDES} Q[i,j]* flmagp[i,j];
39
40
41  end;
```

Figure 5: contenu du fichier PbFluides.mod.txt

On obtient alors les résultats suivants, pour la quantité de Fluide à acheter dans chaque magasin pour minimiser le prix a payer pour les clients :

	$F1$	$F2$
$M1$	2.5	1
$M2$	0.5	1
$M3$	0	1

4.2 Cas particulier 2

Pour le deuxième cas particulier, on ajoute au premier problème des coûts d'éxpeditions à prendre en compte. D'abord, des coûts fixes entre chaque paire demandeur/magasin, mais aussi des coûts variables s'ajoutant en fonction de la quantité de fluide transportée :

	M1	M2	M3
D1	110	90	100
D2	110	90	100

(d) Coûts fixes d'expédition d'un colis entre chaque paire : point de demande, magasin

	M1	M2	M3
D1	10	1	5
D2	2	20	10

(c) Coûts variables d'expédition d'un colis entre chaque paire : point de demande, magasin

Figure 6: Données supplémentaire pour le cas particulier 2

Pour résoudre ce problème on ajoute une nouvelle variable, une matrice F de taille $D_N \times M_N$ où F_{ij} est nul si aucun fluide n'a été acheté au magasin j pour le demandeur i.

On modifie aussi la variable X du problème précedent pour lui ajouter une nouvelle dimension correspondant au demandeur (il n'est en effet plus pertinent de fusionner les demandeurs entre eux car il y a des coûts différents pour chacun d'eux en fonction du magasin dans lequel le fluide est acheté). On définit alors, en plus des constantes et variables définies au problème précedent :

- Cf_{ij} le coût fixe du transport de fluides du magasin j au demandeur i

- Cv_{ij} le coût variable (dépendant de la quantité de fluide transportée) du transport de fluides du magasin j au demandeur i

-$sumfl$ la somme de fluide totale disponible dans tout les magasins

Il faut alors rajouter une contrainte en plus : $\forall i, j \in [1, D_N] \times [1, M_N], \sum_{k=0}^{F_N} X_{ijk} \leq sumfl \times F_{ij}$

Cette contrainte permet de construire la matrice F avec $F - i, j = 1$ si un demandeur à acheter du fluide dans un magasin.

on a alors enfin la fonction a minimiser :

$$\sum_{j=0}^{F_N} \sum_{k=0}^{D_N} (X_{ijk} \times (p_{ij} + Cv_{ki})) + \sum_{i=0}^{D_N} \sum_{j=0}^{M_N} F_{ij} \times Cf_{ij}$$

On peut donc enfin écrire le fichier *ModelFluides2.mod.txt* :

```
############################## Sets ##############################

set FLUIDES;

set DEMANDEURS;

set MAGASINS;

################### Variables ###################

var Q{i in MAGASINS, j in FLUIDES, k in DEMANDEURS}, >=0;

var F{i in DEMANDEURS, j in MAGASINS} ,binary;

################## Constants: Data to load  ########################

param fldem{i in DEMANDEURS, j in FLUIDES};

param flmagst{i in MAGASINS, j in FLUIDES};

param flmagp{i in MAGASINS, j in FLUIDES};

param MDexp{i in DEMANDEURS, j in MAGASINS};

param MDexpc{i in DEMANDEURS, j in MAGASINS};

param sumfl := sum{m in MAGASINS} sum{f in FLUIDES} flmagst[m,f];

################### Constraints ###################


s.t. quantitedem{j in FLUIDES, k in DEMANDEURS}:
  sum{i in MAGASINS} Q[i,j,k] =  fldem[k,j];

s.t. quantitemag{i in MAGASINS,j in FLUIDES}:
  sum{k in DEMANDEURS} Q[i,j,k] <= flmagst[i,j];

s.t. achatmag{i in DEMANDEURS,j in MAGASINS}:
    (sum{k in FLUIDES} Q[j,k,i]) <= sumfl * F[i,j];


###### Objective ######

minimize PrixFluide:
              (sum{i in MAGASINS} sum{j in FLUIDES} sum{k in DEMANDEURS} Q[i,j,k]* (flmagp[i,j] + MDexpc[k,i])) + sum{i in DEMANDEURS} sum{j in MAGASINS} F[i,j] * MDexp[i,j];


end;
```

Figure 7: Modèle a résoudre pour le cas particulier 2

on obtient alors les tableaux de résultats suivant pour chaque client :

	$F1$	$F2$
$M1$	0	0
$M2$	0	0
$M3$	2	0

(a) Résultat optimal pour D1

	$F1$	$F2$
$M1$	1	1
$M2$	0	2
$M3$	0	0

(b) Résultat optimal pour D2

Figure 8: Quantité optimale de fluide a acheter a chaque magasin pour les deux clients de l'exemple

4.3 Cas particulier 3

Le troisiéme cas particulier est un cas de livraison, une société nommé ALPHA doit trouver le chemin optimal que doit prendre un de ses livreurs pour livrer un ensemble de colis à 5 clients sans repasser par l'entrepot. Ce probléme peut nous ramener au probléme du voyageur vu en cours. Ainsi les données fournies pour résoudre ce probléme sont résumé dans une matrice des distances que l'on nomme *dist* représenté par la figure (a).

	$ALPHA$	$C1$	$C2$	$C3$	$C4$	$C5$
$ALPHA$	–	1	1	10	12	12
$C1$	1	–	1	8	10	11
$C2$	1	1	–	8	11	10
$C3$	10	8	8	–	1	1
$C4$	12	10	11	1	–	1
$C5$	12	11	10	1	1	–

(a) matrice des distances *dist*
(magasin ALPHA et 5 clients à livrer)

On a ainsi besoin de deux sets qui correspondent à l'énumération des clients $CLIENTS$ et l'énumérations des clients et du magasin $CLIENTSDEP$, ainsi on a une solution pour N livraisons. Et nous allons représenter le résultat par 2 matrices, une matrice binaire Q, identique à la matrice distance qui définira les trajets que le livreur accompli, et une matrice $ORDRE$ qui contiendra l'ordre des points clients à atteindre pour le trajet du livreur.

Ainsi les contraintes à respecter sont:

Une contrainte pour éviter que le livreur fasse du sur place :

- $\forall i \in [1, N]$, $Q_{i,i} = 0$

Une contrainte sur les colonnes :

- $\forall i, j \in [1, N]$, $\sum_{i=1}^{N} Q_{ij} = 1$

et une contrainte sur les lignes :

- $\forall i, j \in [1, N]$, $\sum_{j=1}^{N} Q_{ij} = 1$

avec $Q_{ij} \in \{0, 1\}$

Ainsi on a donc bien une seul visite par Clients, mais il nous reste un probléme à résoudre, les sous boucles. Ainsi on va utiliser la variable $ORDRE$. Avec 2 contraintes :

La contrainte qui supprime les sous boucles :

-$\forall i \in [1, N], \forall j \in [2, N]$, ($ORDRE_i + (1 - Q_{i,j}) * (NombrSomm + 1)) \geq ORDRE_j + 1$

Et une contrainte qui permet d'avoir des valeurs de 1 à N sur l'ordre :

-$\forall i \in [1, N]$, $ORDRE_i \geq 1$

Enfin, il faut mimimiser la fonction suivante pour obtenir le résultat :

$$\forall i, j \in [1, N], \sum_{i,j=1}^{N} Q_{ij} * dist_{i,j}$$

On peut donc écrire le fichier *ModelLivr.mod.txt* :

```
1  ############################## Model ##############################
2  ############################## Sets ##############################
3  set CLIENTSDEP;
4  set CLIENTS;
5
6  ################## Variables ##################
7  var Q{i in CLIENTSDEP,j in CLIENTSDEP}, binary;
8  var ORDRE{i in CLIENTSDEP}, integer;
9
10 ################## Constants: Data to load  ########################
11 param dist{i in CLIENTSDEP, j in CLIENTSDEP};
12 param NombrSomm := sum{i in CLIENTSDEP} 1;
13
14 ################## Constraints ##################
15 s.t. surplaceinterdit{j in CLIENTSDEP}:
16       Q[j,j] = 0;
17
18 s.t. uniciteCol{j in CLIENTSDEP}:
19    sum{i in CLIENTSDEP} Q[i,j] = 1;
20
21 s.t. uniciteLign{i in CLIENTSDEP}:
22    sum{j in CLIENTSDEP} Q[i,j] = 1;
23
24 s.t. SansSousBoucle{i in CLIENTS,j in CLIENTSDEP}:
25       (ORDRE[i] + (1 - Q[i,j] ) * (NombrSomm+1)) >= ORDRE[j] +1;
26
27 s.t. Ordre{i in CLIENTSDEP}:
28       ORDRE[i] >=1;
29
30 ###### Objective ######
31 minimize distance:
32             sum{i in CLIENTSDEP} sum{j in CLIENTSDEP} Q[i,j]*dist[i,j];
33 end;
```

Figure 10: Modèle a résoudre pour le cas particulier 3

et un fichier de données associé *DataLivr.dat.txt* :

```
# donnees possibles pour le model de distribution
data;

set CLIENTSDEP :=
ALPHA
C1
C2
C3
C4
C5;

set CLIENTS :=
C1
C2
C3
C4
C5;

param dist: ALPHA C1 C2 C3 C4 C5 :=
ALPHA 0 1 1 10 12 12
C1 1 0 1 8 10 11
C2 1 1 0 8 11 10
C3 10 8 8 0 1 1
C4 12 10 11 1 0 1
C5 12 11 10 1 1 0;


end;
```

Figure 11: Données utiliser pour résoudre le cas particulier 3

Ainsi on obtient le résultat suivant :

	$ALPHA$	$C1$	$C2$	$C3$	$C4$	$C5$
$ALPHA$	0	0	1	0	0	0
$C1$	1	0	0	0	0	0
$C2$	0	0	0	0	0	1
$C3$	0	1	0	0	0	0
$C4$	0	0	0	1	0	0
$C5$	0	0	0	0	1	0

(a) Matrice Q

	$Ordre$
$ALPHA$	1
$C1$	2
$C2$	6
$C3$	3
$C4$	4
$C5$	5

(b) Matrice $ORDRE$

pour une $DISTANCE$ finale de 22.