Recherche Opérationelle
Compte-Rendu du TP1

Antoine Girard et Priscilia Gonthier
Groupe L2

Département Sciences du Numérique
2022-2023

Table des matières

Table des figures

1 Exercice 1 : Assemblage

1.1 Introduction

Pour le problème d'assemblage, on considère dans un premier temps que les ouvriers peuvent ne pas avoir achevé la construction d'une voiture à la fin de la semaine. Et dans un second temps, nous nous occuperons du cas où l'on est forcé de finir les véhicules en construstion.

1.2 Cas du Problème Linéaire

On commence par résoudre le **Problème Linéaire** (cf : PbAssemblagePL.lp, PbAssemblagePL.sol)

1.2.1 Choix du fichier

Nous avons choisit pour résoudre ce problème de passer par des fichiers .lp puisque nos variables ne sont que 2 entiers et non pas une matrice. Le fichier .sol, contient la solution de la résolution.

1.2.2 Variables

Les variables de ce problème sont :

1. L = nombre de voitures de Luxe construites.
2. S = nombre de voiture Standard construites.

Nous sommes en programmation linéraire les variables sont donc dans $\mathbb{R}^+$

1.2.3 Fonction objectif

Ainsi pour ce problème on se retrouve avec une fonction objectif que l'on cherche à maximiser représentant le gain de l'entreprise :
Marge = 10000 L + 9000 S

1.2.4 Contraintes

Nous avons déterminé 3 contraintes différentes :

1. Surface_du_parking : 10 L + 20 S $\leq$ 15000
   Permet d'indiquer qu'il n'y a pas plus de voiture construite que la surface du parking ne le permet.

2. Nb_dheure_de_travail : $0,06L + 0,05S \leq 60$
   Permet d'indiquer que les ouvriers ne doivent pas travailler plus que leur quota maximal semainier.
3. Nb_maximal_de_voiture_de_Luxe : $L \leq 800$
   Permet d'indiquer qu'il ne faut pas contruire plus de 800 voitures de luxe pour répondre à la demande des clients.

1.2.5 Résultats

Les résultats indiqués dans la figure 1 sont cohérents puisque si on additionne le nombre d'heures nécessaires pour produire le nombre de voiture souhaité, on tombe bien sur 60h. De plus, si on calcule le gain pour le maximum de voitures standard et le maximum de voiture de luxe, on obtient un gain total moins grand.

```
Problem:
Rows:       3
Columns:    2
Non-zeros:  5
Status:     OPTIMAL
Objective:  Marge = 10285714.29 (MAXimum)

   No.   Row name   St   Activity     Lower bound   Upper bound    Marginal
------ ------------ -- ------------ ------------- ------------- -------------
     1 Surface_du_parking
                    NU         15000                       15000       57.1429
     2 Nb_dheure_de_travail
                    NU            60                          60        157143
     3 Nb_maximal_de_voiture_de_Luxe
                    B         642.857                         800

   No. Column name  St   Activity     Lower bound   Upper bound    Marginal
------ ------------ -- ------------ ------------- ------------- -------------
     1 L            B         642.857             0
     2 S            B         428.571             0
```

FIGURE 1 – Résultat pour le problème d'assemblage, en résolution PL

1.3 Cas du Problème Linéraire en Nombres Entiers

Dans un deuxième temps, nous cherchons à résoudre ce problème avec des variables qui ne peuvent être **que des nombres entiers** (cf : PbAssemblagePLNE.lp, PbAssemblagePLNE.sol).

1.3.1 Choix du fichier

Nous nous plaçons à nouveau dans un fichier **.lp** pour poser ce problème. Le fichier **.sol**, contient la solution de la résolution.

1.3.2 Variables, Fonction objectif et Contraintes

Les conditions, la fonction objectif et les contraintes restent les mêmes, seul le domaine de définition des variables de problème changent. Elles passent de $\mathbb{R}^+$ à $\mathbb{N}$.

1.3.3 Résultats

On obtient encore une fois des résultats cohérents pour les mêmes raisons que précedemment et légèrement plus bas que la résolution précédente du problème (figure 2). Cette diminution est normale puisque nous avons rajouté une contrainte supplémentaire pour la résolution (contrainte de nombre entier).

```
Problem:
Rows:       3
Columns:    2 (2 integer, 0 binary)
Non-zeros:  5
Status:     INTEGER OPTIMAL
Objective:  Marge = 10284000 (MAXimum)

   No.   Row name        Activity     Lower bound   Upper bound
------ ------------    ------------- ------------- -------------
     1 Surface_du_parking
                              14970                         15000
     2 Nb_dheure_de_travail
                                 60                            60
     3 Nb_maximal_de_voiture_de_Luxe
                                645                           800

   No. Column name       Activity     Lower bound   Upper bound
------ ------------    ------------- ------------- -------------
     1 L             *           645             0
     2 S             *           426             0
```

FIGURE 2 – Résultat pour le problème d'assemblage, en résolution PLNE

2 Exercice 2 : Gestion de personnel

2.0.1 Introduction

Pour résoudre le problème de gestion du personnel, nous avons du utiliser **une matrice de variables binaires d'états**. Ainsi, les lignes correspondent au personnel et les colonnes aux travaux. Pour une ligne i et une colonne j données, si le travail correspond à la personne, alors la variable d'état vaut 1, sinon elle vaut 0.

2.0.2 Variables et Choix du fichier

Dans ce problème, on a donc comme variable une matrice de variables d'états :
pour $i \in \text{PERSONNEL}, j \in \text{TRAVAIL}$,

$$\text{affecte}[i,j] = \begin{cases} 1, & \text{si le personnel est affecté au travail} \\ 0, & \text{sinon} \end{cases}$$

Ainsi, pour rendre la résolution et surtout pour rendre la formulation du problème plus simple, nous utilisons un fichier **.mod** pour faire la modélisation, un fichier **.dat** pour renseigner les données d'un problème particulier et un fichier **.sol** contenant la solution du probleme résolu numériquement. (cf : PbPersonnel.mod, PbPersonnel.dat et PbPersonnel.sol)

2.0.3 Fonction objectif

On cherche ici à minimiser le coût de formation du personnel. Ainsi, la fonction objectif est la suivante :

$$\text{CoutTotal} = \sum_{\substack{i \in \text{PERSONNEL} \\ j \in \text{TRAVAIL}}} \text{coutformation}[i,j] \times \text{affecte}[i,j]$$

affecte étant la matrice de variable d'état représentant les affectations.

2.0.4 Contraintes

Nous avons déterminé 2 contraintes différentes dans ce problème :

1. $\forall i \in \text{PERSONNEL}, \sum_{j \in \text{TRAVAIL}} \text{affecte}[i,j] = 1$
   qui permet d'indiquer que chaque personne est obligatoirement affectée à un travail et qu'il est unique
2. $\forall j \in \text{TRAVAIL}, \sum_{i \in \text{PERSONNEL}} \text{affecte}[i,j] = 1$
   qui permet d'indiquer que chaque travail est obligatoirement affecté à une personne et qu'elle est unique

2.0.5 Résultats

Les résultats obtenus sont bien cohérents avec ce que l'on attendait. C'est-à-dire qu'il y a bien chaque travail qui est affecté à une seule personne. Et chaque personne à un seul travail. Pour ce qui est du coût de formation, après avoir fait des tests avec des résultats évidents sur des problèmes de petites dimensions, on remarque que le solveur est correct.

2.0.6 Test réalisé

On a réalisé un test avec 3 travails différents (Coder, Tester et Rédiger) et 3 personnes (Marie, Pierre et Jean). Les coûts de formation ont été écrits pour que la résolution à la main soit facile à faire.

On obtient bien le résultat attendu (figure 3).

```
-----Fin de la résolution -----

Affichage de la solution :
Coût minimal : 5
Affectation : Marie -> Coder (Coût formation : 1)
Affectation : Pierre -> Rediger (Coût formation : 2)
Affectation : Jean -> Tester (Coût formation : 2)
```

FIGURE 3 – Résultat pour le problème d'affectation de personnel à un travail

3 Exercice 3 : E-Commerce

3.1 Cas 1 : Problème Linéaire

3.1.1 Introduction

Pour le problème d'E-commerce, on commence par faire une résolution de **Problème Linéaire** ce qui veut dire que l'on peut prendre une quantité de fluide non-entière dans un magasin.

3.1.2 Variables

Les variables utilisées sont des réels qui vont être représentés par une matrice en deux dimensions. Les lignes représenteront les magasins et les colonnes les fluides.
Ainsi les variables représenteront la quantité de fluide prise dans un magasin.

pour $i \in \text{MAGASIN}, j \in \text{FLUIDE}, \text{quantitefluide}[i,j] \in \mathbb{R}^+$

3.1.3 Choix du fichier

Pour faire la résolution de ce problème nous utilisons un fichier **.mod** ce qui nous permettra de plus facilement rédiger le problème avec des matrices. Les données d'un problème précis seront enregistrées dans un fichier **.dat** et le fichier **.sol** contient la solution du probleme résolu numériquement (cf : PbEcommerce1.mod, PbEcommerce1.dat et PbEcommerce1.sol)

3.1.4 Fonction objectif

L'objectif de ce problème est de minimiser le coût de la commande totale. On en déduit donc la fonction objectif suivante :

$$\text{CoutTotal} = \sum_{\substack{i \in \text{MAGASIN} \\ j \in \text{FLUIDE}}} \text{coutmagasin}[i,j] \times \text{quantitefluide}[i,j]$$

coutmagasin est la matrice de données qui contient le coût de chaque fluide pour chaque magasin.

3.1.5 Contraintes

Nous avons déterminé deux contraintes pour ce problème :

1. $\forall j \in \text{FLUIDE}, \sum_{i \in \text{MAGASIN}} \text{quantitefluide}[i,j] = \sum_{i \in \text{COMMANDE}} \text{fluidecommande[i,j]}$
   Permet de s'assurer que la quantité de fluide fournie est bien la même que la quantité de fluide demandée.
2. $\forall i \in \text{MAGASIN}, \forall j \in \text{FLUIDE}, \text{quantitefluide}[i,j] \leq \text{fluidemagasin[i,j]}$
   Permet de s'assurer que la quantité de fluide prise dans chaque magasin n'excède pas les stocks.

3.1.6 Résultats

Encore une fois, nous avons pour nos tests réalisé un problème facile à résoudre à la main avec seulement 2 fluides, 2 commandes et 3 magasins.

```
param fluidecommande : F1 F2 :=
D1 2 0
D2 1 3;

param fluidemagasin : F1 F2 :=
M1 2.5 1
M2 1 2
M3 2 1;

param coutmagasin : F1 F2 :=
M1 1 1
M2 2 3
M3 3 2;
```

FIGURE 4 – Conditions initiales pour le problème d'e-commerce cas 1

On obtient bien les résultats attendu pour ce problème (figure 5). Le fluide pris dans chaque magasin est bien inférieur au stock du magasin et il y a la bonne quantité de fluide pour les commandes.

```
-----Fin de la résolution -----

Affichage de la solution :
Coût Total : 9.50
Quantité de fluide F1 pris dans le magasin M1 : 2.50
Quantité de fluide F1 pris dans le magasin M2 : 0.50
Quantité de fluide F1 pris dans le magasin M3 : 0.00
Quantité de fluide F2 pris dans le magasin M1 : 1.00
Quantité de fluide F2 pris dans le magasin M2 : 1.00
Quantité de fluide F2 pris dans le magasin M3 : 1.00
```

FIGURE 5 – Résultat pour le problème d'e-commerce cas 1

3.2 Cas 2 : Problème Linéaire en Nombres Entiers

3.2.1 Introduction et choix de fichier

Dans un deuxième temps, nous cherchons à résoudre ce problème avec des variables qui ne peuvent être **que des nombres entiers**, puisque dans ce second cas, les fluides sont considérés comme des produits préconditionnés. Nous avons faits les mêmes choix de fichiers que pour le cas 1 et pour les mêmes raisons (cf : PbEcommerce2.mod, PbEcommerce2.dat et PbEcommerce2.sol).

3.2.2 Variables, Fonction objectif et Contraintes

Les variables, la fonction objectif et les contraintes restent les mêmes, seul le domaine de définition des variables de problème change. Il passe de $\mathbb{R}^+$ à $\mathbb{N}$.

3.2.3 Résultats

On a réalisé des tests sur le même problème que pour le cas 1, en arrondissant à l'entier inférieur les données du problème, afin de pouvoir comparer les deux. On obtient encore une fois des résultats (figure 6) cohérents pour les mêmes raisons que précedemment et un coût légèrement plus élevé, ce qui est normal puisque nous avons rajouté une contrainte supplémentaire pour la résolution (contrainte de nombre entier).

```
-----Fin de la résolution -----

Affichage de la solution :
Coût Total : 10
Quantité de fluide F1 pris dans le magasin M1 : 2
Quantité de fluide F1 pris dans le magasin M2 : 1
Quantité de fluide F1 pris dans le magasin M3 : 0
Quantité de fluide F2 pris dans le magasin M1 : 1
Quantité de fluide F2 pris dans le magasin M2 : 1
Quantité de fluide F2 pris dans le magasin M3 : 1
```

FIGURE 6 – Résultat pour le problème d'e-commerce cas 2

3.3 Cas 3 : Prise en compte du coût des trajets

3.3.1 Introduction

Pour ce problème nous devions prendre en compte les différentes demandes pour s'assurer que les fluides reviennent à un coût minimum pour le demandeur en prennant en compte le coût de chaque fluide dans chaque magasin et le prix de transport entre le magasin et la destination. Comme pour le problème précédent, les fluides sont considérés comme des produits préconditionnés, donc sont entier.
Les formats de fichiers utilisés sont les mêmes que pour le cas 1 et pour les mêmes raisons. (cf : PbEcommerce3.mod, PbEcommerce3.dat et PbEcommerce3.sol)

3.3.2 Variables

Les variables que l'on a choisit sont

1. Une matrice en trois dimensions représentant les quantités de fluide associées aux magasins et aux commandes :
   Pour $i \in \text{MAGASIN}, j \in \text{FLUIDE}, k \in \text{COMMANDE}, \text{quantitefluide}[i, j, k] \in \mathbb{N}$
2. Une matrice en deux dimensions représentant les trajets effectués ou non :

Pour $i \in$ MAGASIN, $k \in$ COMMANDE,

$$\text{trajeteffectue}[i,k] = \begin{cases} 1, & \text{si le trajet est effectué entre le magasin i et la commande k} \\ 0, & \text{sinon} \end{cases}$$

3.3.3 Fonction objectif

L'objectif de ce problème est de minimiser le coût de la commande totale. On en déduit donc la fonction objectif suivante :

$$\begin{aligned} CoutTotal &= \sum_{\substack{i \in \text{MAGASIN} \\ j \in \text{FLUIDE} \\ k \in \text{COMMANDE}}} (\text{coutmagasin}[i,j] \times \text{quantitefluide[i,j,k]}) \\ &+ \sum_{\substack{i \in \text{MAGASIN} \\ k \in \text{COMMANDE}}} (\text{coutfixe[i,k]} \times \text{trajeteffectue[i,k]}) \\ &+ \sum_{\substack{i \in \text{MAGASIN} \\ j \in \text{FLUIDE} \\ k \in \text{COMMANDE}}} (\text{coutvariable[i,k]} \times \text{quantitefluide[i,j,k]}) \end{aligned}$$

3.3.4 Contraintes

Pour ce problème nous avons modélisé au total 3 contraintes, les 2 premières sont les mêmes que pour les cas 1 et 2, elles ont juste étés étendues puisque la matrice de variable est à 3 dimension dans ce cas au lieu de 2 précedemment :

1. $\forall j \in$ FLUIDE, $k \in$ COMMANDE, $\sum_{i \in \text{MAGASIN}} \text{quantitefluide}[i,j,k] = \text{fluidecommande}[k,j]$
   Permet de s'assurer que la quantité de fluide fournie est bien la même que la quantité de fluide commandée.
2. $\forall i \in$ MAGASIN, $j \in$ FLUIDE, $\sum_{k \in \text{COMMANDE}} \text{quantitefluide}[i,j,k] \leq \text{fluidemagasin}[i,j]$
   Permet de s'assurer que la quantité de fluide prise dans chaque magasin n'excède pas les stocks.
3. $\forall i \in$ MAGASIN, $k \in$ COMMANDE,
   Si $\sum_{j \in \text{FLUIDE}} \text{quantitefluide}[i,j,k] > 0$ **Alors**

   trajeteffectue[i,k] = 1

   Sinon

   trajeteffectue[i,k] = 0

   S'il y a un trajet entre le magasin i et la commande k (c'est à dire que la quantité de fluide est non nulle), alors trajeteffectue[i,k] est à 1, à 0 sinon. Si le trajet effectué est nul alors la quantité de fluide est nulle aussi.
   Cela se traduit dans le code par 2 contraintes et l'utilisation d'un big M en résolution

numérique :

$\forall i \in \text{MAGASIN}, k \in \text{COMMANDE}$,

— $\text{trajeteffectue}[i,k] \leq \sum_{j \in FLUIDE} \text{quantitefluide}[i,j,k]$

— $\sum_{j \in FLUIDE} \text{quantitefluide}[i,j,k] \leq \text{trajeteffectue}[i,k] \times M$

Avec **M** qui est calculé à partir des quantités de fluide en stock dans les magasins (égal à la somme de toutes les quantités des magasins)

3.3.5 Résultats

Les résultats (figure 7) sont cohérents car pour chaque demande il prend la bonne quantité de fluide et la prend par ordre croissant du moins cher au plus cher, en prenant en compte le coût du fluide et les coûts de transport.

```
-----Fin de la résolution -----

Affichage de la solution :
Coût Total : 368
Quantité de fluide F1 pris dans le magasin M1  pour la commande D1: 0
Quantité de fluide F1 pris dans le magasin M1  pour la commande D2: 1
Quantité de fluide F1 pris dans le magasin M2  pour la commande D1: 0
Quantité de fluide F1 pris dans le magasin M2  pour la commande D2: 0
Quantité de fluide F1 pris dans le magasin M3  pour la commande D1: 2
Quantité de fluide F1 pris dans le magasin M3  pour la commande D2: 0
Quantité de fluide F2 pris dans le magasin M1  pour la commande D1: 0
Quantité de fluide F2 pris dans le magasin M1  pour la commande D2: 1
Quantité de fluide F2 pris dans le magasin M2  pour la commande D1: 0
Quantité de fluide F2 pris dans le magasin M2  pour la commande D2: 2
Quantité de fluide F2 pris dans le magasin M3  pour la commande D1: 0
Quantité de fluide F2 pris dans le magasin M3  pour la commande D2: 0
```

FIGURE 7 – Résultat pour le problème d'e-commerce cas 3

3.4 Cas 4 : Cas où on cherche à optimiser le trajet d'un livreur

3.4.1 Introduction

Ce problème était le plus compliqué de tous pour nous puisqu'il nous fallait prendre en compte un ordre (ordre de passage du livreur). En effet nous avions fait une première résolution de problème sans prendre en compte l'ordre et la solution faisait téléporter notre livreur ! On a donc décidé d'utiliser une matrice en 3 dimensions plutôt qu'en deux dimensions ce qui va nous permettre de prendre en compte l'ordre de passage et de rajouter des contraintes pour éviter la téléportation.

Les formats de fichiers utilisés sont les mêmes que pour le cas 1 et pour les mêmes raisons. (cf : PbEcommerce4.mod, PbEcommerce4.dat et PbEcommerce4.sol)

3.4.2 Variables

En prenant en compte l'ordre de passage, on se retrouve donc avec la variable suivante :

Pour $i \in$ ORDREPASSAGE, $j \in$ LIEU, $k \in$ LIEU,

$$\text{trajeteffectue}[i,j,k] = \begin{cases} 1, & \text{si le trajet est effectué lors du passage i, du lieu j au lieu k} \\ 0, & \text{sinon} \end{cases}$$

j représentant le lieu de départ, **k** représentant le lieu d'arrivé.

Ainsi, chaque matrice 2 dimensions pour un i précis n'a qu'une seule variable binaire d'état à 1 représentant le ième trajet de j jusqu'à k.

3.4.3 Fonction objectif

On cherche ici à minimiser la distance parcourue par le livreur. On se retrouve donc avec la fonction objectif suivante :

$$\text{DistanceTotal} = \sum_{\substack{i \in \text{ORDREPASSAGE} \\ j \in \text{LIEU} \\ k \in \text{LIEU}}} \text{distance}[j,k] \times \text{trajeteffectue[i,j,k]}$$

3.4.4 Contraintes

Pour ce problème nous avons déterminé un total de 6 contraintes différentes :

1. $\forall j \in$ LIEU, $\sum_{i \in \text{ORDREPASSAGE}} \text{trajeteffectue[i,j,j]} = 0$
   Permet d'éviter les trajets entre un lieu et ce même lieu.
2. $\forall d \in$ DEPART, $\sum_{k \in \text{LIEU}} \text{trajeteffectue[1,d,k]} = 0$
   Permet de garantir qu'initialement le livreur va partir de l'entrepot de stockage.
3. $\forall j \in$ LIEU, $\sum_{\substack{i \in ORDREPASSAGE \\ k \in \text{LIEU}}} \text{trajeteffectue[i,j,k]} = 1$
   Permet de garantir que l'on ai qu'un seul trajet partant de chaque lieu.
4. $\forall k \in$ LIEU, $\sum_{\substack{i \in ORDREPASSAGE \\ j \in \text{LIEU}}} \text{trajeteffectue[i,j,k]} = 1$
   Permet de garantir que l'on ai qu'un seul trajet ayant comme destination un lieu k.
5. $\forall i \in$ ORDREPASSAGE, $\sum_{\substack{j \in LIEU \\ k \in \text{LIEU}}} \text{trajeteffectue[i,j,k]} = 1$
   Permet de garantir que l'on ait qu'un seul trajet par passage.
6. $\forall i \in [\![1, n-1]\!], j \in$ LIEU, $\sum_{l \in \text{LIEU}} \text{trajeteffectue[i+1,j,l]} = \sum_{k \in \text{LIEU}} \text{trajeteffectue[i,k,j]}$
   Permet de s'asurer que le livreur parte du lieu k (destination du trajet i) lors de son

trajet i+1, et donc éviter les téléportations, avec **n** le nombre de lieu à visiter (donné dans le fichier de données).

3.4.5 Résultats

Pour ce qui est des résultats nous obtenons bien un résultat cohérent car le livreur fait bien sa boucle de livraison sans se téléporter et en ne passant qu'une seule fois par chaque lieu.

```
-----Fin de la résolution -----

Affichage de la solution :
Distance minimale : 22
Trajet 1 : ALPHA -> C2 (Distance : 1)
Trajet 2 : C2 -> C3 (Distance : 8)
Trajet 3 : C3 -> C5 (Distance : 1)
Trajet 4 : C5 -> C4 (Distance : 1)
Trajet 5 : C4 -> C1 (Distance : 10)
Trajet 6 : C1 -> ALPHA (Distance : 1)
```

FIGURE 8 – Résultat pour le problème d'e-commerce cas 4