Modélisation + Résolution de PL/PLNE avec le solveur GLPK

Léo Meissner et Lisa Weisbecker

Département Sciences du Numérique - Deuxième année
2022-2023

Table des matières

Table des figures

1 Introduction

Ce TP a pour but d'implanter et résoudre avec GLPK les modèles que nous proposons pour chaque problème. Pour chaque problème, nous :

1. expliquerons le choix de variable de décision et son domaine de définition.
2. justifierons la pertinence du format utilisé.
3. étudierons les résultats ainsi que la cohérence des résultats obtenus.

2 Problème Assemblage

Pour représenter ce problème nous utilisons 2 variables, Q1 représentant le nombre de voitures de luxe assemblées et Q2 le nombre de voitures standard assemblées, car on souhaite connaître le nombre de voitures de chaque type qu'il faut produire pour maximiser les bénéfices.
D'après l'énoncé, $Q1, Q2 \in \mathbf{N^2}$.

Ensuite, nous avons choisi le format lp, car les contraintes sont peu nombreuses et ne se répètent pas. Par conséquent, ce format est suffisant pour répondre au problème.

La fonction objective à maximiser est :

$$Benefice = 10000Q1 + 9000Q2$$

Et les contraintes que nous avons récupérées de l'énoncé sont les suivantes :

$$CapaciteduTravail : 0.06Q1 + 0.05Q2 <= 60$$

$$CapacitedeSurface : 10Q1 + 20Q2 <= 15000$$

$$CapaciteAssemblage : Q1 <= 800$$

Avec ce modèle, nous obtenons un bénéfice maximum de 10284000 avec $Q1 = 645$ et $Q2 = 426$
Ce résultat nous semble cohérent car chaque contrainte est respectée et la contrainte des heures de travail est saturée (cohérent pour obtenir un bénéfice maximal).

3 Problème Gestion de personnel

Cette fois-ci, nous avons choisi une matrice de taille N a valeur binaire comme variable de décision nommée matricedecision dans notre modèle. Ainsi, les lignes de la matrice représentent les personnes et les colonnes les tâches. Par conséquent, il suffit de regarder la ligne de notre matrice pour voir une tâche réalisée par une personne et une colonne pour voir la personne qui réalise une tâche.
De plus, la matrice est de valeurs binaires, nous avons défini qu'une tache ne peut être réalisée qu'une seule fois et par une seule personne.

Ensuite, nous avons choisi le format mod pour modéliser ce problème, car les contraintes de chaque ligne et de chaque colonne sont similaires. Ici nous pouvons réfléchir en terme de personnes et de taches et non pas seulement en fonction de paramètres, ce n'est pas possible avec le format lp.

La fonction objective à minimiser est :

$$CoutTotal = \sum_{i=1}^{N} \sum_{j=1}^{N} matricedecision(i,j) * matricecout(i,j)$$

Et les contraintes que nous avons récupérées de l'énoncé sont les suivantes :
Une tâche par personne et une personne par tâche

$$RespectUnePersonneUneTache = \forall i \sum_{j=1}^{N} matricedecision(i,j) = 1$$

$$RespectUneTacheUnePersonne = \forall j \sum_{i=1}^{N} matricedecision(i,j) = 1$$

Par exemple pour la matrice :
matricecout : $\begin{bmatrix} 10 & 1 & 10 \\ 15 & 10 & 1 \\ 1 & 15 & 15 \end{bmatrix}$
Nous obtenons bien la matrice de décison :

matricedecision : $\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$
Par conséquent, notre modèle est cohérent.

4 Problème E-commerce cas1

Dans notre modèle, le 1er cas d'application au e-commerce, la variable de décision dépend de 3 choses, les fluides, les stocks et les magasins. Par conséquent, notre variable de décision est une matrice en 3 dimensions, la première représente les magasins, la deuxième, les fluides et la troisième les demande. Nous trouvons cette modélisation pertinente, car nous avons besoin de ces 3 dimensions pour minimiser le coutTotal.
De plus comme les fluides sont des quantités réels, nous avons $\forall i, \forall j, \forall k, d_{i,j,k} \in \mathbf{R}^+$ avec d un élément de la matrice de décision.

Nous avons choisi le format mod pour les mêmes raisons que le problème précédent.
La fonction objective à minimiser est :

$$CoutTotal = \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{N} decision(i,j,k) * coutFluides(i,j)$$

Les contraintes pour ce problème sont les suivantes : pour chaque magasin, les stocks doivent être respectés dans la décision et chaque demande doit être prise en compte par la matrice de décision. Nous avons modélisé comme suit :

$$RespectStocks = \forall i, j \sum_{k=1} decision(i,j,k) <= stocksFluides(i,j)$$

$$RespectDemandes = \forall k, j \sum_{i=1} decision(i,j,k) = demandesFluides(k,j)$$

Avec ce modèle et les données de l'énoncé, nous obtenons un coût total minimal de 9,5 comme le montre la figure 1.

Pour vérifier la cohérence de notre modèle, nous avons calculé le coût minimal de ce jeu de donnée à la main et nous trouvons également 9,5.

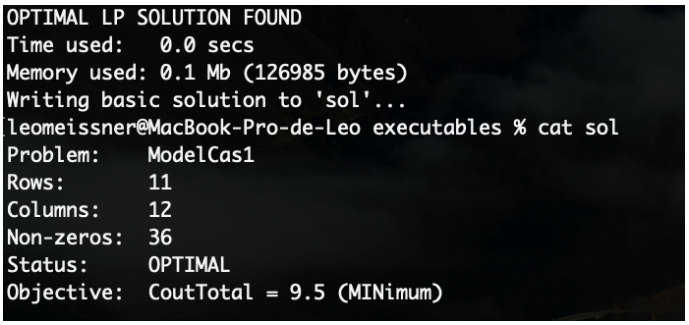

FIGURE 1 – Résultat cas 1

5 Problème E-commerce cas2

Ici, le seul changement avec le cas 1 est que les valeurs de notre matrice de décision sont entières au lieu d'être réelles. Nous avons donc changer notre modèle pour prendre cette modification en compte et nous avons changer le 2.5 en 2 dans le jeu de données. Le reste du modèle est inchangé.

Avec ce modèle et les données de l'énoncé, nous obtenons un coût total minimal de 10 comme le montre la figure 2.

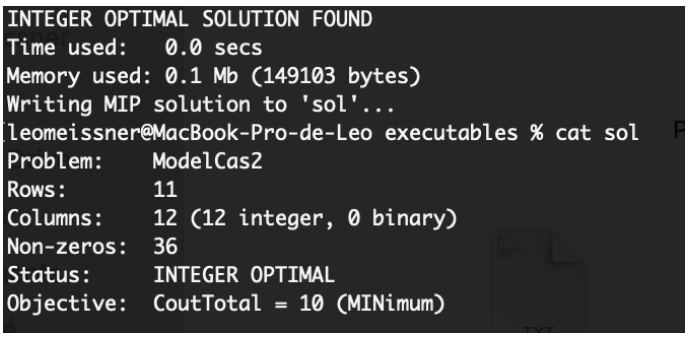

FIGURE 2 – Résultat cas 2

Pour vérifier la cohérence de notre modèle, nous avons calculé le coût minimal de ce jeu de donnée à la main et nous trouvons également 10.

6 Problème E-commerce cas3

Dans notre modèle,le 3^{eme} cas d'application au e-commerce notre 1^{ere} variable de décision dépend de 3 choses, les produits, les stocks et les magasins. Par conséquent, notre variable de décision est une matrice en 3 dimensions, la première représente les magasins, la deuxième, les produits et la troisième les demandes.
De plus comme les produits sont des quantités réels, nous avons $\forall i, \forall j, \forall k, d_{i,j,k} \in \mathbf{R}^+$ avec d un élément de la 1^{ere} variable de décision.

Or cette variable ne suffit pas pour modéliser les coutsFixes, il nous faut introduire une 3^{eme} variable de décision qui s'active lorsqu'un magasin délivre au moins 1 produit. Nous avons donc choisi une deuxième variable qui est la liste binaire des magasins. Une valeur vaut 1 si un magasin délivre au moins 1 produit, 0 sinon.

Nous avons choisi le format mod pour les mêmes raisons que les autres problèmes de ce type.

La fonction objective à minimiser est :

$$CoutTotal = \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N} decision(i,j,k) * coutVariable(i,j) + \sum_{i=1}^{N}\sum_{j=1}^{N} impact(j,i) * coutFixes(j,i)$$

Les contraintes pour ce problème sont les suivantes : Pour chaque magasin, les stocks doivent être respectés dans la décision. Pour chaque demande doit être prise en compte par la matrice de décision. Le coût fixe est pris en compte si au moins un produit du magasin est utilisé. Nous les avons modélisées comme suit :

$$RespectStocks = \forall i,j \sum_{k=1} decision(i,j,k) <= stocksFluides(i,j)$$

$$RespectDemandes = \forall k,j \sum_{i=1} decision(i,j,k) = demandesFluides(k,j)$$

$$RespectImpact1 = \forall k,i \sum_{j=1} decision(i,j,k) >= impact(k,i)$$

$$RespectImpact2 = \forall k,i \sum_{j=1} decision(i,j,k) <= bigM * impact(k,i)$$

Pour respecter la dernière contrainte, nous avons introduit un paramètre supplémentaire, le bigM qui est un entier valant 10000. La double contrainte que nous introduisons fonctionne alors comme suit : D'un côté si un magasin envoie au moins un produit alors $\forall k,i \sum_{j=1} decision(i,j,k) >= 1 = impact(k,i)$ et $\forall k,i \sum_{j=1} decision(i,j,k) <= 10000 * impact(k,i) = 10000$. Comme notre paramètre bigM est bien supérieur à $\sum_{j=1} decision(i,j,k)$, les 2 contraintes sont respectés. D'un autre côté, si le magasin n'envoie aucun produit chaque quantité vos 0 donc les contraintes sont respectés.

Avec ce modèle et les données de l'énoncé, nous obtenons un coût total minimal de 354 comme le montre la figure 3.

```
INTEGER OPTIMAL SOLUTION FOUND
Time used:   0.0 secs
Memory used: 0.2 Mb (175584 bytes)
Writing MIP solution to 'sol'...
leomeissner@MacBook-Pro-de-Leo executables % cat sol
Problem:    ModelCas3
Rows:       23
Columns:    18 (18 integer, 6 binary)
Non-zeros:  78
Status:     INTEGER OPTIMAL
Objective:  CoutTotal = 354 (MINimum)
```

FIGURE 3 – Résultat cas 3

7 Problème E-commerce cas4

Ce cas est un problème similaire au problème du voyageur de commerce qui est un cas que nous avons évoqué en cours. Dans un premier temps, nous avons pensé à la modéliser de manière analogue au problème de gestion de personnel, car il faut se rendre d'un client à un autre en passant une seule fois chez chaque client. Donc, notre première variable de décision était une matrice binaire où une case (i,j) à la valeur 1 lorsqu'on se rend au client j depuis le client i.

Les contraintes étaient les suivantes :

$$RespectAutovisite = \forall i decision(i,i) = 0$$

$$RespectLigne = \forall i \sum_{j=1}^{N} decision(i,j) = 1$$

$$RespectColonne = \forall j \sum_{i=1}^{N} decision(i,j) = 1$$

Cette modélisation n'a pas fonctionnée, car nous ne prenons pas en compte le lien entre chaque client : si le livreur s'est rendu au client (i,j), il doit repartir de celui-ci pour son prochain voyage. Donc nous avons ajouté les contraintes suivantes :

$$RespectSousBoucle = \forall i \neq' ALPHA' decision2(i) >= 2$$

$$RespectSousBoucle2 = \forall i \neq' ALPHA', j \neq' ALPHA'$$

$$decision2(i) - decision2(j) + 1 <= (card(BATIMENTS) - 1) * (1 - decision(i,j))$$

Ainsi, le coût minimal pour effectuer le trajet donné dans l'énoncé est de 22, cf figure 4. De plus, nous avons testé avec des valeurs évidentes et notre modèle choisissait le bon trajet.

```
INTEGER OPTIMAL SOLUTION FOUND
Time used:   0.0 secs
Memory used: 0.2 Mb (213879 bytes)
Writing MIP solution to 'sol'...
leomeissner@MacBook-Pro-de-Leo executables % cat sol
Problem:    ModelCas4
Rows:       49
Columns:    41 (41 integer, 36 binary)
Non-zeros:  178
Status:     INTEGER OPTIMAL
Objective:  CoutTotal = 22 (MINimum)
```

FIGURE 4 – Résultat cas 4

8 Conclusion

Ce tp nous a permis d'en apprendre plus sur les différents formats de modélisation ainsi que de leur pertinence au regard du problème qui nous ai posé, nous l'avons donc trouvé intéressant.