Rapport TP1 : Modélisation et Résolution de PL/PLNE avec le solveur GLPK

DUBOIS Lucas et PEYREMORTE Romain

Département Sciences du Numérique - Seconde année - Recherche Opérationnelle
2022-2023

Table des matières

Table des figures

1 Introduction du rapport

Dans ce projet, nous nous sommes intéressé à la résolution de différents problèmes d'optimisation de société tels que la minimisation des trajets de livraison, et l'optimisation des coûts pour des productions ou pour des formations.

Nous avons pour cela modéliser les différents problèmes en format lp et/ou en format gmpl, pour une solution plus générique. Puis nous avons obtenu les résultats à l'aide de GLPK (GNU Linear Programming Kit).

Nous avons dans ce rapport détaillé les axes de réfléxion pour parvenir à modéliser chaque problème, ainsi que la résolution réalisée grâce à l'outil GLPK.

L'ensemble des programmes cités sont fournis avec ce rapport.

2 Assemblage

Notre premier cas d'étude est la production de deux modèles de voitures : une dite de luxe, notée L, et une standard, notée S. L'usine produisant ces voitures souhaite savoir comment répartir le travail entre les deux modèles de voiture pour maximiser la marge totale après la vente de ses véhicules sur une semaine.

2.1 Modélisation

L'équipe d'ouvrier travaille au maximum 60 heures par semaine. Une fois qu'une voiture est produite, elle est stockée sur un parking de 15000m^2 qui est vidé à la fin de la semaine.

Pour ce qui est des informations sur les différents modèles, nous les avons notées dans le tableau suivant :

Modèle	Durée production 100 véhicules	Espace occupé	Max production par semaine	Marge par voiture	Production par semaine
L	6	10m^2	800	10000	n_L
S	5	20m^2	∞	9000	n_S

On prend donc n_L et n_S les variables de notre problème qui correspondent à la production de chaque modèle durant une semaine. Dans un premier cas, nous réaliserons l'étude à l'aide d'un modèle de Problème Linéaire, où on a donc $(n_L, n_S) \in \mathbb{R}^2$. Ainsi, on a le problème suivant :

$$\min_{(n_L,n_S)\in\mathbb{R}^2} MargeTot(n_L, n_S) = 10000n_L + 9000n_S$$
$$\begin{cases} \frac{6}{100}n_L + \frac{5}{100}n_S \leq 60 \\ 10n_L + 20n_S \leq 15000 \\ n_L \leq 800 \end{cases}$$

Cependant, il est difficile de vendre des morceaux de voiture à la fin de chaque semaine, c'est pourquoi il vaudrait mieux que la production de voiture soit discrète, donc que $(n_L, n_S) \in \mathbb{N}^2$. On passe alors sur un modèle de Problème Linéaire à Nombre Entier. On a ainsi le problème suivant :

$$\min_{(n_L,n_S)\in\mathbb{N}^2} MargeTot(n_L, n_S) = 10000n_L + 9000n_S$$
$$\begin{cases} \frac{6}{100}n_L + \frac{5}{100}n_S \leq 60 \\ 10n_L + 20n_S \leq 15000 \\ n_L \leq 800 \end{cases}$$

Par rappport au premier cas, ce second change juste le domaine de définition des variables.

2.2 Résolution

Pour cette première étude, nous avons décidé de retranscrire le modèle dans un fichier de type *.lp*. Les fichiers sont *assemblageReel.lp.txt* pour le cas Réel (PL) et *assemblageEntier.lp.txt* pour le cas Entier (PLNE). Il reprend les données de l'énoncé. Ce format de fichier n'impose qu'un seul cas d'application : on ne peut pas appliquer ce modèle à des cas avec plus de voitures ou d'autres données. Cela peut être possible grâce au format *.mod* que l'on utilisera dans les prochaines études.

Les figures 1 et 2 correspondent aux solutions renvoyées grâce à la résolution. On trouve un maximum de 10285714,29€ dans le cas réel et un maximum de 10284000€ dans le cas naturel, ce qui parait logique avec le maximum réel plus grand que le naturel entier, car le cas naturel est inclu dans le cas réel.

Le fait de passer dans le cas des entiers naturels se justifie en observant qu'il semble bien impossible de produire (et d'arrêter la production avant le week-end) 0,857 de voiture...

3 Gestion de personnel

Dans cette deuxième étude, nous nous intéressons à l'affectation de personnels à des postes de travail. Chaque personne ne peut occuper qu'un poste et un poste ne peut être occupé que par une personne. De plus le personnel a des compétences différentes, ce qui fait que la formation et son coût sont différents entre chaque personnes à chaque postes. Le but est de répartir le personnel aux différents postes en minimisant le coût total de formation de chaque travailleur.

3.1 Modélisation

On se place dans un cas où il y a N personnes et N postes à pourvoir (N travaux à réaliser). Comme données pour notre problème, nous avons la matrice c de taille N qui correspond aux coûts de formation de chaque personne à chaque poste : $c_{i,j}$ correspond au coût de formation de la personne i au poste j.

Nous avons décidé d'utiliser une variable de a (pour affectation) de type matrice de taille N défini sur $\{0;1\}$: la personne i travaille au poste j si $a_{i,j} = 1$. Si elle n'y travaille pas, alors $a_{i,j} = 0$. Ainsi, en utilisant le modèle d'une PLNE, on obtient le problème suivant :

$$\min_{a_{i,j} \in \{0;1\}} CoutTot(a_{i,j}) = \sum_{i=1,j=1}^{N,N} a_{i,j} c_{i,j}$$

$$\begin{cases} \forall i \in [\![1;N]\!], \sum_{j=1}^{N} a_{i,j} = 1 \\ \forall j \in [\![1;N]\!], \sum_{j=1}^{N} a_{i,j} = 1 \end{cases}$$

3.2 Résolution

Pour cette étude, nous avons réalisé le modèle de résolution sous le fichier gestionPersonnelle.mod.txt. Grâce à cette modélisation (et ce format de fichier), on peut appliquer différents cas, avec différents jeux de données. On a alors créé deux cas : un cas basique avec deux postes et deux personnes *gestionPersonnellebasique.dat.txt*, et un cas plus complexe avec 10 postes et 10 personnes *gestionPersonnellecomplexe.dat.txt*.

Dans le cas basique, on a pris deux personnes qui sont respectivement mieux adaptés au premier et au deuxième poste. Le résultat obtenu est dans la figure 3. On obtient l'objectif escompté.

Dans le cas plus complexe, on a augmenté le nombre de personnes ainsi que créé une ambiguïté avec un poste où deux personnes ont le même coût de formation, avec une différence sur un autre poste. Pour les autres personnes, chacun à un poste où son coût de formation est plus faible que le reste des différents postes. L'objectif obtenu attribue bien à chaque personne le poste ayant le coût le plus faible.

4 Applications en optimisation pour l'e-commerce

Dans cette partie, nous allons étudier différents cas sur l'e-commerce avec des problèmes d'affectation de commandes à différents magasins dans le but de minimiser les coûts financiers et/ou environnementaux.

4.1 Cas particulier 1

Dans ce premier cas, nous commençons par une série de commandes réalisée sur des fluides qui se trouvent dans différents magasins.

4.1.1 Modélisation

Dans ce cas, nous travaillons sur des fluides, les quantités manipulées sont donc des réels pouvant être à virgule (nous ne nous limitons pas qu'aux entiers dans ce premier cas).

Nous avons comme informations les quantités commandées par chaque client que l'on notera $d_{i,k}$ (pour demande) avec i le fluide demandé et k le client commandant le fluide. Nous avons aussi les stocks de chaque magasin pour chaque fluide que l'on notera $s_{i,j}$ avec i le fluide en stock dans le magasin j. Enfin, nous avons aussi les coûts unitaires de livraison de chaque magasin pour chaque produit que l'on notera $cu_{i,j}$ avec i le fluide livré et j le magasin.

Nous avons décidé de définir la variable quantité $q_{i,j,k} \in \mathbb{R}_+$ qui correspond à la quantité de fluide i livrée par le magasin j au client k. Dans notre problème, nous devons faire attention à ce que chaque commande soit respectée et que les magasins ne soient pas à court de fluide. Ainsi, en appliquant le modèle d'un PL, on obtient le problème suivant :

$$\min_{q_{i,j,k} \in \mathbb{R}_+} CoutTot(q_{i,j,k}) = \sum_{i=1,j=1}^{N_{magasins},N_{fluides}} \left(\sum_{k=1}^{N_{demandes}} q_{i,j,k} \right) cu_{i,j}$$

$$\begin{cases} \forall i \in [\![1; N_{fluides}]\!], \forall k \in [\![1; N_{demandes}]\!], \sum_{j=1}^{N_{magasins}} q_{i,j,k} = d_{i,k} \\ \forall i \in [\![1; N_{fluides}]\!], \forall j \in [\![1; N_{magasins}]\!], \sum_{k=1}^{N_{demandes}} q_{i,j,k} \leq s_{i,j} \end{cases}$$

4.1.2 Résolution

Pour ce premier cas, nous avons réalisé le modèle de résolution sous le fichier *optiECommerceCas1.mod.txt*. Nous avons appliqué ce modèle sur le jeu de donnée *optiECommerceCas1.dat.txt* présent dans l'énoncé. Le résultat obtenu est dans les figures 4 et 5. On obtient comme résultat de minimisation un coût total de 9.5.

4.2 Cas particulier 2

Le cas 2 est similaire au premier cas, on passe juste de la vente et de la livraison de fluide à celles de produits.

4.2.1 Modélisation

Ce changement implique que les quantités deviennent discrètes : maintenant, on $q_{i,j,k} \in \mathbb{N}$. On a donc maintenant un Problème Linéaire à Nombre Entier :

$$\min_{q_{i,j,k} \in \mathbb{N}} CoutTot(q_{i,j,k}) = \sum_{i=1,j=1}^{N_{magasins},N_{produits}} \left(\sum_{k=1}^{N_{demandes}} q_{i,j,k} \right) cu_{i,j}$$

$$\begin{cases} \forall i \in [\![1; N_{produits}]\!], \forall k \in [\![1; N_{demandes}]\!], \sum_{j=1}^{N_{magasins}} q_{i,j,k} = d_{i,k} \\ \forall i \in [\![1; N_{produits}]\!], \forall j \in [\![1; N_{magasins}]\!], \sum_{k=1}^{N_{demandes}} q_{i,j,k} \leq s_{i,j} \end{cases}$$

4.2.2 Résolution

Dans le deuxième cas, nous avons réalisé le modèle de résolution sous le fichier *optiECommerceCas2.mod.txt*. Nous avons appliqué ce modèle sur le même jeu de donnée présent dans l'énoncé. Le résultat obtenu est dans les figures 6 et 7. Le minimum trouvé de 10 est logique car il est supérieur au minimum obtenu dans le cas précédent. En effet, comme on a pu le dire dans l'étude de l'assemblage de voiture, l'ensemble des entiers auxquel on se limite dans la PLNE est compris dans l'ensemble des réels utilisé dans la PL, et donc le minimum de PLNE peut-être le minimum de PL, mais pas l'inverse. Il aurait été surprenant de trouver un meilleur résultat avec un modèle PLNE qu'avec un modèle PL lorsque l'on applique au même jeu de données.

4.3 Cas particulier 3

Ce nouveau cas s'intéresse aux coûts financiers d'expédition des colis envoyés par les magasins aux clients avec un colis par magasin par client. Ces coûts financiers d'expédition correspondent aux coûts fixes du colis, auquel on ajoute les coûts variables qui dépendent de la quantité de produit dans le colis. Ces deux coûts sont indépendants de la nature des produits.

4.3.1 Modélisation

Nous avons donc avec ce nouveau cas deux coûts qui s'ajoutent : les coûts fixes que l'on notera $cf_{j,k}$ avec j le magasin et k le client livré, et les coûts variables $cv_{j,k}$ avec j le magasin et k le client livré.

Pour simplifier la manipulation et la compréhension, nous garderons la variable $q_{i,j,k}$ même si ce cas ne nécessite pas de faire varier le produit (i inutile).

Nous avons rencontré des difficultés avec les coûts fixes car ils apparaissent de façon binaire. En effet, nous avons des coûts fixe pour chaque magasin, qui ne sont comptabilisés que si des produits sont livrés à un client. C'est pourquoi nous avons décidé d'ajouter la variable $cop_{j,k} \in \{0;1\}$ (pour "colis ou pas") où $cop_{j,k} = 1$ si le magasin j doit livrer au client k et $cop_{j,k} = 0$ sinon. On a donc :

$$\begin{cases} \forall j \in [\![1; N_{magasins}]\!], \forall k \in [\![1; N_{demandes}]\!], \sum_{i=1}^{N_{produits}} q_{i,j,k} \geq 1 \Rightarrow cop_{j,k} = 1 \ (1) \\ \forall k \in [\![1; N_{demandes}]\!], \sum_{i=1}^{N_{produits}} q_{i,j,k} = 0 \Rightarrow cop_{j,k} = 0 \ (2) \end{cases}$$

Pour pouvoir générer ces implications, nous avons décidé d'utiliser la méthode *Big M* qui permet d'avoir l'implication de l'égalité en résultat binaire de façon linéaire avec l'utilisation d'un M très grand. Nous devons alors avoir :

$$\forall j \in [\![1; N_{magasins}]\!], \forall k \in [\![1; N_{demandes}]\!], \sum_{i=1}^{N_{produits}} q_{i,j,k} \leq M$$

On a donc choisi :

$$M = \sum_{i=1,j=1}^{N_{produits},N_{magasins}} s_{i,j}$$

Ainsi en appliquant la méthode *Big M*, nous avons les deux contraintes suivantes :

$$\begin{cases} \forall j \in [\![1; N_{magasins}]\!], \forall k \in [\![1; N_{demandes}]\!], cop_{j,k} - \left(\sum_{i=1}^{N_{produits}} q_{i,j,k} \right) \leq 0 \ (3) \\ \forall j \in [\![1; N_{magasins}]\!], \forall k \in [\![1; N_{demandes}]\!], \left(\sum_{i=1}^{N_{produits}} q_{i,j,k} \right) - \left(\sum_{i=1,j=1}^{N_{produits},N_{magasins}} s_{i,j} \right) cop_{j,k} \leq 0 \ (4) \end{cases}$$

Ainsi l'inégalité (3) permet d'avoir l'implication (2), et l'inégalité (4) l'implication (1). En applicant dont un modèle PLNE, on obtient le problème suivant :

$$\min_{q_{i,j,k} \in \mathbb{N}, cop_{j,k} \in \{0;1\}} CoutTot(q_{i,j,k}, cop_{j,k}) = \sum_{j=1,k=1}^{N_{magasins},N_{demandes}} cop_{j,k} cf_{i,j} + \sum_{j=1,k=1}^{N_{magasins},N_{demandes}} \left(\sum_{i=1}^{N_{produits}} q_{i,j,k} \right) cv_{j,k}$$

$$\begin{cases} \forall j \in [\![1; N_{magasins}]\!], \forall k \in [\![1; N_{demandes}]\!], cop_{j,k} - \left(\sum_{i=1}^{N_{produits}} q_{i,j,k} \right) \leq 0 \\ \forall j \in [\![1; N_{magasins}]\!], \forall k \in [\![1; N_{demandes}]\!], \left(\sum_{i=1}^{N_{produits}} q_{i,j,k} \right) - \left(\sum_{i=1,j=1}^{N_{produits},N_{magasins}} s_{i,j} \right) cop_{j,k} \leq 0 \end{cases}$$

4.3.2 Résolution

Pour le troisième cas, nous avons réalisé le modèle de résolution sous le fichier *optiECommerceCas3.mod.txt*. Nous avons appliquer ce modèle sur le même jeu de donnée que précemment en ajoutant celui des coûts fixes et variables présent dans l'énoncé du projet. Le résultat obtenu est dans les figures 8 et 9. On obtenons le résultat voulu.

4.4 Cas particulier 4

Dans ce cas, nous nous intéressons à la livraison vers différents clients à partir d'un magasin. Le livreur doit réaliser une boucle : c'est à dire parcourir tout les clients à la suite avant de revenir au magasin. En sachant les distances séparant les différents lieux (magasin et clients), nous voulons minimiser la distance parcourue par le livreur durant sa tournée. Ce problème reprend celui du *voyageur de commerce*.

4.4.1 Modélisation et Résolution

La modélisation du problème sous forme de PLNE ne nous est pas apparu facilement, nous avons dû réaliser le modèle à taton avec différents tests de résolution pour voir les résultats obtenus. C'est pourquoi nous traiterons la résolution en même temps que la modélisation.

Le problème est de trouver quel est le parcours entre chaque lieu qui permet d'obtenir la distance minimale parcouru. C'est pourquoi, pour représenter le choix de parcours entre deux lieux, nous avons décidé d'utiliser la matrice da (pour départ/arrivée) de taille $taille \times taille$ où $taille$ correspond au nombre de lieux à visiter, c'est à dire aux nombre de clients incrémenté de 1 avec le magasin de départ. La matrice da est définie telle que :

1. $(i,j) \in [\![1; taille]\!]^2, da_{i,j} \in \{0;1\}$
2. $da_{i,j} = 1$ si le livreur durant son trajet part du lieu i pour aller au lieu j
3. $da_{i,j} = 0$ lorsque le livreur ne réalise pas le trajet de i à j

Les conditions doivent alors assurer que chaque lieu ne soit visité qu'une seule fois, c'est à dire que le livreur arrive et part de chaque lieu qu'une fois. De plus, on a comme information la matrice des distances d entre chaque lieu telle que $d_{i,j}$ est la distance entre le lieu i et le lieu j.

Avec cela, on a une première version du problème sous modèle PLNE :

$$\min_{da_{i,j} \in \{0;1\}} DistanceTot(da_{i,j}) = \sum_{i=1,j=1}^{taille,taille} da_{i,j} d_{i,j}$$

$$\begin{cases} \forall i \in [\![1; taille]\!], \sum_{j=1}^{taille} da_{i,j} = 1 \\ \forall j \in [\![1; taille]\!], \sum_{j=1}^{taille} da_{i,j} = 1 \end{cases}$$

En appliquant les données de l'énoncé et ce modèle, on trouve une distance totale de 0 mètre avec :

$$da = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

On a donc le livreur reste au même endroit pour chaque trajet, ce qui n'a pas de sens. Il manque donc une condition pour empêcher cela. C'est pourquoi nous avons défini le modèle suivant :

$$\min_{da_{i,j}\in\{0;1\}} DistanceTot(da_{i,j}) = \sum_{i=1,j=1}^{taille,taille} da_{i,j}d_{i,j}$$

$$\begin{cases} \forall i \in [\![1;taille]\!], \sum_{j=1}^{taille} da_{i,j} = 1 \\ \forall j \in [\![1;taille]\!], \sum_{j=1}^{taille} da_{i,j} = 1 \\ \sum_{i=1}^{taille} da_{i,i} = 0 \end{cases}$$

Avec ce modèle nous obtenons une distance totale de 6 mètres ainsi que la matrice :

$$da = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

On remarque alors des cycles dans la tournée : le livreur part du magasin, se rend à C1, puis à C2, puis retourne au magasin, puis part de C3 pour aller à C4 puis aller à C5 et retourner à C3. Le livreur s'est donc comme téléporté entre le magasin et C3.

Pour éviter ce genre d'événement, il faut forcer l'idée du cycle qui commence par le magasin, passe par tous les clients, puis finit par le magasin. Pour répondre à cette attente, nous avons défini la variable *op* (pour ordre de passage) qui est un vecteur de taille *taille* tel que :

1. $i \in [\![1;taille]\!], op_i \in [\![1;taille]\!]$
2. op_i correspond à l'ordre de visite du lieu i (le lieu i sera le op_i^{eme} lieu visité)

On peut voir à travers cette définition une construction récursive de *op* :

1. Initialisation : $op_1 = 1$
2. Héritage : $i \in [\![1;taille]\!], j \in [\![2;taille]\!], i \neq j, x_{i,j} = 1 \implies t_i + 1 = t_j$

Pour pouvoir réaliser l'initialisation, on peut définir un ensemble uniquement composé du magasin pour avoir la condition :

$$\forall i \in \{1\}(ou\ MAGASIN), op_i = 1$$

Pour l'héritage, on peut utiliser la méthode *Big M* pour avoir l'implication avec $M = taille$, ce qui permet d'avoir les conditions suivantes :

$$\begin{cases} \forall i \in [\![1;taille]\!], \forall j \in [\![1;taille]\!]\backslash\{1\}, i \neq j, op_j - (op_i + 1) \geq -taille \times (1 - da_{i,j}) \\ \forall i \in [\![1;taille]\!], \forall j \in [\![1;taille]\!]\backslash\{1\}, i \neq j, op_j - (op_i + 1) \leq taille \times (1 - da_{i,j}) \end{cases}$$

On a ainsi le modèle final suivant :

$$\min_{da_{i,j}\in\{0;1\}} DistanceTot(da_{i,j}) = \sum_{i=1,j=1}^{taille,taille} da_{i,j}d_{i,j}$$

$$\begin{cases} \forall i \in [\![1;taille]\!], \sum_{j=1}^{taille} da_{i,j} = 1 \\ \forall j \in [\![1;taille]\!], \sum_{j=1}^{taille} da_{i,j} = 1 \\ \sum_{i=1}^{taille} da_{i,i} = 0 \\ \forall i \in \{1\}, op_i = 1 \\ \forall i \in [\![1;taille]\!], \forall j \in [\![1;taille]\!]\backslash\{1\}, i \neq j, op_j - (op_i + 1) \geq -taille \times (1 - da_{i,j}) \\ \forall i \in [\![1;taille]\!], \forall j \in [\![1;taille]\!]\backslash\{1\}, i \neq j, op_j - (op_i + 1) \leq taille \times (1 - da_{i,j}) \end{cases}$$

Avec ce modèle, nous obtenons, en appliquant les données du sujet écrites dans le fichier *optiECommerceCas4.dat.txt*, une distance total de 22.

Comme la matrice des données est symétrique parau niveau de sa diagonale ($d_{i,j} = d_{j,i}$), on peut obtenir deux chemins possibles. Par exemple, en se limitant aux conditions suffisantes de la modèlisation, on trouve la matrice de départ/arrivée ainsi que le vecteur d'ordre de passage suivante :

$$da = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

$$op = \begin{pmatrix} 1 \\ 2 \\ 6 \\ 3 \\ 4 \\ 5 \end{pmatrix}$$

Mais en ajoutant la condition nécessaire mais pas utile (mise en commentaire dans le fichier) :

$$\sum_{i=1}^{taille} op_i = \frac{taille \times (taille + 1)}{2}$$

On trouve comme résultat 22 en distance totale mais le chemin inverse :

$$da = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

$$op = \begin{pmatrix} 1 \\ 6 \\ 2 \\ 5 \\ 4 \\ 3 \end{pmatrix}$$

En modifiant les données d'entrée en augmentant par exemple une valeur sans garder la symétrie (par exemple pour aller de C5 à C2, le coût devient 11 au lieu de 10 car il y a une pente entre les deux (et donc il est plus facile, et moins coûteux de descendre que de monter) par exemple), avec ou sans la condition, on trouve toujours le même résultat (22), avec le même chemin (la deuxième matrice).

Ce qui montre que la résolution avec notre modèle peut être influencé par l'ajout ou non d'une contrainte avec des données symétriques, mais reste bon dans tout les cas.

Pour tester encore plus notre modèle, nous avons appliqué un jeux de données avec 16 lieux : les données sont celles du voyagee d'Ulysse de dim 16, qui semble être un exemple récurrent dans la résolution du problème du voyage de commerce. Ces données sont trouvable dans le fichier *optiECommerceCas4Ulysse16.dat.txt*.

Malgrès un temps de résolution beaucoup plus important (de >1s à 30s), on obtient une distance totale minimale de 6859 qui est le résultat attendu associé à ce problème. Cela prouve que notre modèle peut être appliqué à d'autres cas, tant que l'on défini bien les données selon le modèle,

même de taille plus grande. Cependant, la complexité de résolution peut peut-être remettre en cause son utilisation dans des cas très grand, le modèle n'est peut-être pas le plus optimisé pour ce genre de problème.

5 Conclusion

Ainsi, ce projet nous a permis de mettre en place et de tester concrètement des problèmes d'optimisation et de les résoudre avec un solveur GLPK qui s'est avéré efficace.

Ce fut intéressant de transformer des problèmes du langage français en problème linéaire, de découvrir des méthodes permettant de transformer des implications d'égalités en inégalités, même si ce fut laborieux.

6 Annexe

Problem :
Rows : 5
Columns : 2
Non-zeros : 7
Status : OPTIMAL
Objective : MargeTot = 10285714.29 (MAXimum)
No. Row name St Activity Lower bound Upper bound Marginal
——— ———— – ————- ————- ————- ————-
1 TempsMaxTravail
NU 60 60 157143
2 EspaceParkingMax
NU 15000 15000 57.1429
3 MaxProductionL
B 642.857 800
4 r.11 B 642.857 0
5 r.12 B 428.571 0
No. Column name St Activity Lower bound Upper bound Marginal
——— ———— – ————- ————- ————- ————-
1 nL B 642.857 0
2 nS B 428.571 0
Karush-Kuhn-Tucker optimality conditions :
KKT.PE : max.abs.err = 7.11e-15 on row 1 max.rel.err = 5.87e-17 on row 1 High quality
KKT.PB : max.abs.err = 0.00e+00 on row 0 max.rel.err = 0.00e+00 on row 0 High quality
KKT.DE : max.abs.err = 1.82e-12 on column 1 max.rel.err = 9.09e-17 on column 1 High quality
KKT.DB : max.abs.err = 0.00e+00 on row 0 max.rel.err = 0.00e+00 on row 0 High quality
End of output

FIGURE 1 – Résultat en réel flottant du problème Assemblage

Problem :
Rows : 3
Columns : 2 (2 integer, 0 binary)
Non-zeros : 5
Status : INTEGER OPTIMAL
Objective : MargeTot = 10284000 (MAXimum)
No. Row name Activity Lower bound Upper bound
——— ———— ————- ————- ————-
1 TempsMaxTravail
60 60
2 EspaceParkingMax
14970 15000
3 MaxProductionL
645 800
No. Column name Activity Lower bound Upper bound
——— ———— ————- ————- ————-
1 nL * 645 0
2 nS * 426 0
Integer feasibility conditions :
KKT.PE : max.abs.err = 0.00e+00 on row 0 max.rel.err = 0.00e+00 on row 0 High quality
KKT.PB : max.abs.err = 0.00e+00 on row 0 max.rel.err = 0.00e+00 on row 0 High quality
End of output

FIGURE 2 – Résultat en entier naturel du problème Assemblage

```
Problem : gestionPersonnelle
Rows : 5
Columns : 4 (4 integer, 4 binary)
Non-zeros : 12
Status : INTEGER OPTIMAL
Objective : CoutTot = 33 (MINimum)
No. Row name Activity Lower bound Upper bound
------ ------------ ------------- ------------- -------------
1 RespectAttributionPersonne[Michel]
1 1 =
2 RespectAttributionPersonne[Patrice]
1 1 =
3 RespectAttributionPostes[Boulanger] 1 1 =
4 RespectAttributionPostes[Charcutier]
1 1 =
5 CoutTot 33
No. Column name Activity Lower bound Upper bound
------ ------------ ------------- ------------- -------------
1 affectation[Michel,Boulanger]
* 1 0 1
2 affectation[Michel,Charcutier]
* 0 0 1
3 affectation[Patrice,Boulanger]
* 0 0 1
4 affectation[Patrice,Charcutier]
* 1 0 1
Integer feasibility conditions :
KKT.PE : max.abs.err = 0.00e+00 on row 0 max.rel.err = 0.00e+00 on row 0 High quality
KKT.PB : max.abs.err = 0.00e+00 on row 0 max.rel.err = 0.00e+00 on row 0 High quality
End of output
```

FIGURE 3 – Résultat basique de la gestion de personnelle

Problem : optiECommerceCas1
Rows : 11
Columns : 12
Non-zeros : 36
Status : OPTIMAL
Objective : CoutTot = 9.5 (MINimum)
No. Row name St Activity Lower bound Upper bound Marginal
——— ———— – ———— ———— ———— ————
1 RespectCommande[F1,D1]
NS 2 2 = 2
2 RespectCommande[F1,D2]
NS 1 1 = 2
3 RespectCommande[F2,D1]
B 0 -0 =
4 RespectCommande[F2,D2]
NS 3 3 = 3
5 RespectStocks[F1,M1]
NU 2.5 2.5 -1
6 RespectStocks[F1,M2]
B 0.5 1
7 RespectStocks[F1,M3]
B 0 2
8 RespectStocks[F2,M1]
NU 1 1 -2
9 RespectStocks[F2,M2]
B 1 2
10 RespectStocks[F2,M3]
NU 1 1 -1
11 CoutTot B 9.5

FIGURE 4 – Résultat du cas 1 de l'e-commerce

No. Column name St Activity Lower bound Upper bound Marginal
——— ———————— — ————————— ————————— ————————— —————————
1 quantite[F1,M1,D1]
B 2 0
2 quantite[F1,M2,D1]
NL 0 0 < eps
3 quantite[F1,M3,D1]
NL 0 0 1
4 quantite[F1,M1,D2]
B 0.5 0
5 quantite[F1,M2,D2]
B 0.5 0
6 quantite[F1,M3,D2]
NL 0 0 1
7 quantite[F2,M1,D1]
NL 0 0 3
8 quantite[F2,M2,D1]
NL 0 0 3
9 quantite[F2,M3,D1]
NL 0 0 3
10 quantite[F2,M1,D2]
B 1 0
11 quantite[F2,M2,D2]
B 1 0
12 quantite[F2,M3,D2]
B 1 0
Karush-Kuhn-Tucker optimality conditions :
KKT.PE : max.abs.err = 0.00e+00 on row 0 max.rel.err = 0.00e+00 on row 0 High quality
KKT.PB : max.abs.err = 0.00e+00 on row 0 max.rel.err = 0.00e+00 on row 0 High quality
KKT.DE : max.abs.err = 0.00e+00 on column 0 max.rel.err = 0.00e+00 on column 0 High quality
KKT.DB : max.abs.err = 0.00e+00 on row 0 max.rel.err = 0.00e+00 on row 0 High quality
End of output

FIGURE 5 – Résultat du cas 1 de l'e-commerce (suite)

Problem : optiECommerceCas2
Rows : 11
Columns : 12 (12 integer, 0 binary)
Non-zeros : 36
Status : INTEGER OPTIMAL
Objective : CoutTot = 10 (MINimum)
No. Row name Activity Lower bound Upper bound
——— ———— ————- ————- ————-
1 RespectCommande[P1,D1]
2 2 =
2 RespectCommande[P1,D2]
1 1 =
3 RespectCommande[P2,D1]
0 -0 =
4 RespectCommande[P2,D2]
3 3 =
5 RespectStocks[P1,M1]
2 2.5
6 RespectStocks[P1,M2]
1 1
7 RespectStocks[P1,M3]
0 2
8 RespectStocks[P2,M1]
1 1
9 RespectStocks[P2,M2]
1 2
10 RespectStocks[P2,M3]
1 1
11 CoutTot 10

FIGURE 6 – Résultat du cas 2 de l'e-commerce

```
No. Column name Activity Lower bound Upper bound
------ ------------ ------------ ------------ ------------
1 quantite[P1,M1,D1]
* 1 0
2 quantite[P1,M2,D1]
* 1 0
3 quantite[P1,M3,D1]
* 0 0
4 quantite[P1,M1,D2]
* 1 0
5 quantite[P1,M2,D2]
* 0 0
6 quantite[P1,M3,D2]
* 0 0
7 quantite[P2,M1,D1]
* 0 0
8 quantite[P2,M2,D1]
* 0 0
9 quantite[P2,M3,D1]
* 0 0
10 quantite[P2,M1,D2]
* 1 0
11 quantite[P2,M2,D2]
* 1 0
12 quantite[P2,M3,D2]
* 1 0
Integer feasibility conditions :
KKT.PE : max.abs.err = 0.00e+00 on row 0 max.rel.err = 0.00e+00 on row 0 High quality
KKT.PB : max.abs.err = 0.00e+00 on row 0 max.rel.err = 0.00e+00 on row 0 High quality
End of output
```

FIGURE 7 – Résultat du cas 2 de l'e-commerce (suite)

Problem : optiECommerceCas3
Rows : 23
Columns : 18 (18 integer, 6 binary)
Non-zeros : 78
Status : INTEGER OPTIMAL
Objective : CoutTot = 354 (MINimum)

No.	Row name	Activity	Lower bound	Upper bound
1	RespectCommande[P1,D1]	2	2	=
2	RespectCommande[P1,D2]	1	1	=
3	RespectCommande[P2,D1]	0	-0	=
4	RespectCommande[P2,D2]	3	3	=
5	RespectStocks[P1,M1]	1		2.5
6	RespectStocks[P1,M2]	0		1
7	RespectStocks[P1,M3]	2		2
8	RespectStocks[P2,M1]	1		1
9	RespectStocks[P2,M2]	2		2
10	RespectStocks[P2,M3]	0		1
11	RespectPasColis[M1,D1]	0	-0	
12	RespectPasColis[M1,D2]	-1	-0	
13	RespectPasColis[M2,D1]	0	-0	
14	RespectPasColis[M2,D2]	-1	-0	
15	RespectPasColis[M3,D1]	-1	-0	
16	RespectPasColis[M3,D2]	0	-0	
17	RespectColis[M1,D1]	0	-0	
18	RespectColis[M1,D2]	-7.5	-0	
19	RespectColis[M2,D1]	0	-0	
20	RespectColis[M2,D2]	-7.5	-0	
21	RespectColis[M3,D1]	-7.5	-0	
22	RespectColis[M3,D2]	0	-0	
23	CoutTot	354		

FIGURE 8 – Résultat du cas 3 de l'e-commerce

```
No. Column name Activity Lower bound Upper bound
------ ------------ ------------ ------------ ------------
1 quantite[P1,M1,D1]
* 0 0
2 quantite[P1,M2,D1]
* 0 0
3 quantite[P1,M3,D1]
* 2 0
4 quantite[P1,M1,D2]
* 1 0
5 quantite[P1,M2,D2]
* 0 0
6 quantite[P1,M3,D2]
* 0 0
7 quantite[P2,M1,D1]
* 0 0
8 quantite[P2,M2,D1]
* 0 0
9 quantite[P2,M3,D1]
* 0 0
10 quantite[P2,M1,D2]
* 1 0
11 quantite[P2,M2,D2]
* 2 0
12 quantite[P2,M3,D2]
* 0 0
13 colisOuPas[M1,D1]
* 0 0 1
14 colisOuPas[M1,D2]
* 1 0 1
15 colisOuPas[M2,D1]
* 0 0 1
16 colisOuPas[M2,D2]
* 1 0 1
17 colisOuPas[M3,D1]
* 1 0 1
18 colisOuPas[M3,D2]
* 0 0 1
Integer feasibility conditions :
KKT.PE : max.abs.err = 0.00e+00 on row 0 max.rel.err = 0.00e+00 on row 0 High quality
KKT.PB : max.abs.err = 0.00e+00 on row 0 max.rel.err = 0.00e+00 on row 0 High quality
End of output
```

FIGURE 9 – Résultat du cas 3 de l'e-commerce (suite)