Rapport du TD1/TP1 : Recherche Opérationelle

Modélisation + Résolution de PL/PLNE avec le solveur GLPK

Ikram Berroug & Myriam ROBBANA

Decembre 2024

Table des matières

Liste des tableaux

1 Assemblage

Dans le problème d'assemblage des vélos cargos et standards présenté, l'objectif est de savoir comment répartir le travail entre les deux modèles de vélos pour que la marge totale soit la plus grande possible.

1.1 Modélisation

Il faudrait définir une **fonction objectif** qui maximise la marge totale, qui est la somme des marges sur chaque type de vélo : 700 € pour chaque vélo cargo V_c et 300 € pour chaque vélo standard V_s.

Les **contraintes** prennent en compte plusieurs facteurs : d'une part, la limite du temps de travail hebdomadaire (60 heures), d'autre part, la capacité du parking (1500 m^2) qui impose une contrainte sur le nombre total de vélos stationnés, et enfin, une restriction sur le nombre maximal de vélos cargos à assembler (700 vélos).

Nous avons donc décider par la suite de définir deux **variables** V_c et V_s représentant respectivement le nombre de vélos cargos et standards à assembler, et leur domaine est défini comme étant l'ensemble des réels positifs $\mathbb{R}+^2$, car ces quantités doivent être non négatives.

Voici, ci-dessous, la modélisation choisie pour notre problème PL.

Variables	**Fonction objectif**
V_c : nombre de vélos cargo V_s : nombre de vélos standards	$\max 700 \cdot V_c + 300 \cdot V_s$
Contraintes	**Domaine**
$\frac{6}{100} \cdot V_c + \frac{5}{100} \cdot V_s \leq 60$ $2.5 \cdot V_c + V_s \leq 1500$ $V_c \leq 700$	$(V_c, V_s) \in \mathbb{R}+^2$

Table 1 – Modélisation du problème d'Assemblage dans le cas PL.

Nous allons maintenant nous placer dans un cadre dans lequel notre problème se modélise par PLNE. Ainsi, le domaine de définition de nos variables Vs et Vc change, et ces variables deviennent donc des entiers naturels.

Variables	**Fonction objectif**
V_c : nombre de vélos cargo V_s : nombre de vélos standards	$\max 700 \cdot V_c + 300 \cdot V_s$
Contraintes	**Domaine**
$\frac{6}{100} \cdot V_c + \frac{5}{100} \cdot V_s \leq 60$ $2.5 \cdot V_c + V_s \leq 1500$ $V_c \leq 700$	$(V_c, V_s) \in \mathbb{N}^2$

Table 2 – Modélisation du problème d'Assemblage dans le cas PLNE.

1.2 Résultats obtenus

```
Problem:
Rows: 3
Columns: 2 (2 integer, 0 binary)
Non-zeros: 5
Status: INTEGER OPTIMAL
Objective: Benefice = 438400 (MAXimum)

   No. Row name Activity Lower bound Upper bound
------ ------------ ------------- ------------- -------------
     1 ContrainteTemps
                                5992 6000
     2 ContrainteSurface
                                1500 1500
     3 ContrainteQuantiteVC
                                 232 700

   No. Column name Activity Lower bound Upper bound
------ ------------ ------------- ------------- -------------
     1 VC * 232 0
     2 VS * 920 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
```

Analyse des résultats :

On remarque dans le fichier de résultats généré que les différentes contraintes imposées par notre modèle sont bien respectées. En effet, la **ContrainteTemps** est satisfaite car $5992 < 6000$. De plus, la **ContrainteSurface** est également respectée, étant donné que la surface occupée est exactement de $1500\,m^2$. Enfin, la **ContrainteQuantitéVC** est respectée, avec une valeur de V_c inférieure à 700 vélos.

Par ailleurs, les quantités V_c et V_s retenues sont respectivement 232 vélos et 920 vélos. En appliquant ces valeurs à la formule permettant de calculer les bénéfices : on a bien $700.VC + 300.VS = 438400$, ce qui est le résultat donné pour la maximisation.

Le fichier solution obtenu pour cette partie semble donc cohérent.

Les fichiers fournis pour cette partie sont : `PbAssemblage.lp.txt` et `solAssemblage.sol.txt` .

2 Affectation avec prise en compte des préférences

Dans ce problème, l'objectif est de préparer le planning d'une équipe de N personnes. Durant la journée, il y a N tâches à effectuer. Chaque tâche doit être affectée exactement une fois et chaque personne doit effectuer exactement une tâche. Chaque membre de l'équipe a fait part de ses préférences quant aux différentes tâches, qui se traduit par un score de préférence noté sur 10.

Ainsi, $c(i, j)$ correspond au score de préférence de la personne Pi pour la tâche Tj.

2.1 Modélisation

L'objectif de la manageuse est de déterminer la meilleure affectation possible, c'est-à-dire attribuer à chaque personne la tâche qui lui est la plus favorable.Ainsi, la **fonction objectif** consistera à maximiser la somme des scores de préférence associés aux tâches attribuées à chaque individu.

Les **contraintes** à respecter sont les suivantes : chaque tâche doit être affectée une seule fois, et chaque personne doit se voir attribuer exactement une tâche.

Les **variables** que nous avons considérées comme les plus appropriées pour ce problème sont les booléens x_{ij}, qui indiquent si la tâche Tj est attribuée à la personne Pi. Ces variables prendront les valeurs 0 (pour false) et 1 (pour true), ce qui permet de formuler mathématiquement les contraintes en fonction de ces variables..

Voici la modélisation retenue pour notre problème.

Variables	**Fonction objectif**
x_{ij} : booléen indiquant si la tâche Tj est effectuée par la personne Pi	$\max \sum_{i,j} x_{ij} \cdot c_{ij}$
Contraintes	**Domaine**
$\sum_i x_{ij} = 1, \quad \forall i \in \{1, \ldots, N\}$ $\sum_j x_{ij} = 1, \quad \forall j \in \{1, \ldots, N\}$	$x_{ij} \in \{0, 1\}, \quad \forall i, j \in \{1, \ldots, N\}$

Table 3 – Modélisation du problème d'affectation avec prise en compte des préférences.

2.2 Résultats obtenus

Afin de tester notre modèle, nous avons pris la matrice de préférence ci-dessous.

	T1	T2	T3
P1	4	8	6
P2	7	1	3
P3	5	5	7

Table 4 – Matrice de préférences ($N = 3$).

```
Problem:
Rows: 6
Columns: 9 (9 integer, 9 binary)
Non-zeros: 18
Status: INTEGER OPTIMAL
Objective: BeneficeTotal = 22 (MAXimum)

   No. Row name Activity Lower bound Upper bound
------ ------------ ------------- ------------- -------------
     1 RespectUneTachePourUnePersonne(T1)
                                1 1 =
     2 RespectUneTachePourUnePersonne(T2)
                                1 1 =
     3 RespectUneTachePourUnePersonne(T3)
                                1 1 =
     4 RespectUnePersonnePourUneTache(P1)
                                1 1 =
     5 RespectUnePersonnePourUneTache(P2)
                                1 1 =
     6 RespectUnePersonnePourUneTache(P3)
                                1 1 =

   No. Column name Activity Lower bound Upper bound
------ ------------ ------------- ------------- -------------
     1 X(P1,T1) * 0 0 1
     2 X(P2,T1) * 1 0 1
     3 X(P3,T1) * 0 0 1
     4 X(P1,T2) * 1 0 1
     5 X(P2,T2) * 0 0 1
```

```
     6 X(P3,T2) * 0 0 1
     7 X(P1,T3) * 0 0 1
     8 X(P2,T3) * 0 0 1
     9 X(P3,T3) * 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
```

Analyse des résultats :

On remarque dans le fichier de résultats généré que les différentes contraintes imposées par notre modèle sont bien respectées. En effet, nous avons exactement une tâche qui est allouée à chaque personne. On peut observer dans le fichier que pour chaque tâche T_i, on a à la fois le Lower bound et le Upper Bound qui valent 1 dans la contrainte **RespectUneTachePourUnePersonne(Ti)**. De manière analogue, on a également le Lower et le Upper bound qui valent 1 pour toute personne P_i dans la contrainte **RespectUnePersonnePourUneTache(Pi)**.

Par ailleurs, les différentes valeurs allouées au tableau X nous permettent de voir que P_1 est allouée à la tâche T_2, P_2 à la tâche T_1 et enfin P_3 obtient la tâche T_3. Ainsi, en tenant compte de ces informations, on peut calculer le **BeneficeTotal** à partir de la **Matrice de préférences** : $8 + 7 + 7 = 22$. Ce Résultat étant bien conforme à la valeur donnée dans le fichier.

Le fichier solution obtenu pour cette partie semble donc cohérent.

Les fichiers fournis pour cette partie sont : `PbAffectationAvecPreferences.lp.txt` et `solAffectationAvecPreferences.sol.txt` .

3 Application en optimisation pour l'e-commerce

L'affectation des commandes des clients aux magasins représente l'une des problématiques d'optimisation majeures dans le secteur de l'e-commerce. En effet, il est crucial d'optimiser les coûts liés à la livraison des colis, à la préparation des commandes et à la gestion des stocks.

Dans cette section, nous nous concentrerons spécifiquement sur le problème d'affectation des commandes (cas particuliers **1.2** et **2.2**) ainsi que sur l'optimisation des tournées de véhicules pour différents magasins d'une même franchise (cas particulier **2**), avec pour objectif de minimiser les coûts.

3.1 Cas particulier 1.1

3.1.1 Modélisation

L'objectif dans ce cas particulier est de minimiser le coût total des demandes en fluide provenant de différentes commandes, en tenant en compte les coûts unitaires spécifiques à chaque magasin d'origine. Ainsi, la **fonction objectif** consistera à minimiser la somme des coûts de distribution des fluides, en fonction des stocks disponibles dans chaque magasin.

Les **contraintes** à respecter sont les suivantes : chaque commande ne peut recevoir plus de fluide que ce que le stock du magasin peut fournir, et toutes les demandes en fluides doivent être entièrement satisfaites.

Les **variables** que nous avons choisies comme les plus pertinentes pour ce problème sont les réels x_{ijk} représentant la quantité de fluide Fj provenant du magasin Mk allouée à la demande Di.

Voici la modélisation retenue pour notre problème de programmation linéaire.

Variables	**Fonction objectif**
x_{ijk} : quatité de fluide Fj provenant du magasin Mk allouée à la demande Di	min $\sum_{k \in \text{MAGASIN}, j \in \text{FLUIDE}} \sum_{i \in \text{DEMANDE}} x_{ijk} \cdot C_{kj}$
Contraintes	**Domaine**
$\sum_{i \in \text{DEMANDE}} x_{ijk} \leq S_{kj}, \quad \forall j \in \text{FLUIDE}, \forall k \in \text{MAGASIN}$ $\sum_{k \in \text{MAGASIN}} x_{ijk} = D_{ij}, \quad \forall i \in \text{DEMANDE}, \forall j \in \text{FLUIDE}$	$x_{ij} \in \{0, 1\}, \quad \forall i, j \in \{1, \ldots, N\}$

TABLE 5 – Modélisation du problème de e-commerce pour le cas 1.1

Les variables C, S et D mentionnées ci-dessus représentent les éléments suivants :
C_{kj} : coût unitaire du fluide Fj du magasin Mk
S_{kj} : Stock du fluide Fj dans le magasin Mk
D_{ij} : quantité du fluide Fj demandés par la commande Di

3.1.2 Résultats obtenus

Afin de tester notre modèle, nous avons choisi des exemples des tables du TD représentées ci-dessous :

	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

TABLE 6 – Représentation des fluides demandés, stocks et coûts par magasin.

```
Problem:
Rows: 10
Columns: 12
Non-zeros: 24
Status: OPTIMAL
Objective: CoutTotal = 9.5 (MINimum)

   No. Row name St Activity Lower bound Upper bound Marginal
------ ------------ -- ------------- ------------- ------------- -------------
     1 ContrainteStock(F1,M1)
                    NU 2.5 2.5 -1
     2 ContrainteStock(F1,M2)
                    B 0.5 1
     3 ContrainteStock(F1,M3)
                    B 0 2
     4 ContrainteStock(F2,M1)
                    NU 1 1 -2
     5 ContrainteStock(F2,M2)
                    B 1 2
     6 ContrainteStock(F2,M3)
                    NU 1 1 -1
     7 ContrainteDemande(D1,F1)
                    NS 2 2 = 2
     8 ContrainteDemande(D1,F2)
                    B 0 -0 =
     9 ContrainteDemande(D2,F1)
                    NS 1 1 = 2
    10 ContrainteDemande(D2,F2)
                    NS 3 3 = 3

   No. Column name St Activity Lower bound Upper bound Marginal
------ ------------ -- ------------- ------------- ------------- -------------
     1 X(D1,F1,M1) B 2 0
     2 X(D2,F1,M1) B 0.5 0
     3 X(D1,F1,M2) NL 0 0 < eps
     4 X(D2,F1,M2) B 0.5 0
```

```
    5 X(D1,F1,M3) NL 0 0 1
    6 X(D2,F1,M3) NL 0 0 1
    7 X(D1,F2,M1) NL 0 0 3
    8 X(D2,F2,M1) B 1 0
    9 X(D1,F2,M2) NL 0 0 3
   10 X(D2,F2,M2) B 1 0
   11 X(D1,F2,M3) NL 0 0 3
   12 X(D2,F2,M3) 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
```

Analyse des résultats :

On remarque dans le fichier de résultats généré que les différentes contraintes imposées par notre modèle sont bien respectées. En effet, pour chaque fluide i et chaque magasin j, on la **ContrainteStock(**F_i**,**M_j**)** qui s'assure que la quantité de fluide F_i relevée dans chaque magasin (Dans la colonne **Activity**) est toujours inférieure à la quantité total de fluide F_i dans le magasin. De manière analogue, on voit bien dans **ContrainteDemande** qu'on prend répond toujours exactement à la demande en terme de quantité de fluide.

Par ailleurs, les différentes valeurs allouées au tableau X nous permettent de voir les quantités prélevées pour chaque fluide, pour chaque magasin et pour chaque demande. Ainsi, on tenant compte de ces informations, on peut calculer le **CoutTotal** à partir de **la matrice des coûts unitaires par magasin d'origine** ce qui nous donne : $2x1 + 0.5x1 + 0.5x2 + 1x1 + 1x3 + 1x2 = 9.5$. Ce Résultat étant bien conforme à la valeur donnée dans le fichier.

Le fichier solution obtenu pour cette partie semble donc cohérent.

Les fichiers fournis pour cette partie sont : `ModelECommerce1.mod.txt`, `DataECommerce1All.dat.txt`, `PbECommerce1.lp.txt` et `SolECommerce1.sol.txt`.

3.2 Cas particulier 1.2

Dans ce problème, on prend en compte les coûts d'expédition des colis, composés d'un coût fixe (émissions polluantes de base) et d'un coût variable (lié à la charge transportée). Le but sera donc de modifier la modélisation pour intégrer ces coûts et résoudre le problème avec les données fournies.

3.2.1 Modélisation

Dans le modèle qui suit, les **variables** sont définies pour représenter la quantité de colis allouée à chaque demande ainsi que les décisions binaires indiquant si un magasin approvisionne une demande donnée.

La **fonction objectif** a été choisie pour minimiser les coûts totaux, en tenant compte à la fois des coûts fixes (liés à l'activation d'un magasin pour une demande) et des coûts variables (associés au transport des colis). La multiplication par y_{mdc} permet d'inclure les coûts fixes uniquement lorsque le magasin M_m est effectivement utilisé pour répondre à la demande D_d.

Les **contraintes** garantissent la satisfaction de la demande pour chaque type de colis et le respect des capacités de stock des magasins. L'inégalité $y_m dc$ ¡= $x_m dc$ permet de s'assurer que lorsque la valeur de la quantité $x_m dc$ est nulle, le booléen $y_m dc$ sera mise à 0. De manière analogue, la deuxième inégalité permet de s'assurer que dès que $x_m dc$ est positive, la variable booléen $y_m dc$ sera mise à 1.

Dans notre cas, les quantité $x_m dc$ sont des entiers naturels car on est dans un PLNE. D'où le domaine défini ci-dessous.

Voici la modélisation retenue pour notre problème de programmation linéaire.

Variables	**Fonction objectif**
x_{mdc} : Quantité de colis C_c provenant du magasin M_m allouée à la demande D_d y_{mdc} : Variable booléenne qui vaut 1 si on a pris le colis C_c provenant du magasin M_m pour la demande D_d, 0 sinon.	$\min \sum_{m\in\text{MAGASIN},c\in\text{COLIS},d\in\text{DEMANDE}} y_{mdc} \cdot (C_{\text{fixe},dm} + C_{\text{variable},dm})$
Contraintes	**Domaine**
$\sum_{d\in\text{DEMANDE}} x_{mdc} \leq S_{mc}, \quad \forall c \in \text{COLIS}, \forall m \in \text{MAGASIN}$ $\sum_{m\in\text{MAGASIN}} x_{mdc} = CD_{dc}, \quad \forall d \in \text{DEMANDE}, \forall c \in \text{COLIS}$ $x_{mdc} \leq y_{mdc}, \quad \forall m \in \text{MAGASIN}, c \in \text{COLIS}, d \in \text{DEMANDE}$ $x_{mdc} \leq y_{mdc} \cdot (S_{mc} + CD_{dc}), \quad \forall m \in \text{MAGASIN}, c \in \text{COLIS}, d \in \text{DEMANDE}$	$x_{mdc} \geq 0, \quad \forall m, d, c$ $y_{mdc} \in \{0, 1\}, \quad \forall m, d, c$

TABLE 7 – Modélisation du problème du problème de e-commerce pour le cas 1.2

Les variables CD et S mentionnées ci-dessus représentent les éléments suivants :

CD_{dc} : colis c demandé par la commande d
S_{mc} : stocks de colis c présents dans le magasin m

3.2.2 Résultats obtenus

Afin de tester notre modèle, nous avons choisi des exemples de coûts fixes, variables, colis demandé et stock de colis propre à notre TD. Les différentes tables sont détaillées ci-dessous.

	C1	C2
D1	2	0
D2	1	3

(a) Colis demandés par commande

	C1	C2
M1	3	1
M2	1	2
M3	2	1

(b) Stock de colis par magasin

	M1	M2	M3
D1	110	90	100
D2	110	90	100

(c) Coûts fixes d'expédition

	M1	M2	M3
D1	10	1	5
D2	2	20	10

(d) Coûts variables d'expédition

TABLE 8 – Paramètres pour l'expédition et la gestion des colis.

```
Problem:
Rows: 34
Columns: 24 (24 integer, 12 binary)
Non-zeros: 72
Status: INTEGER OPTIMAL
Objective: CoutTotal = 435 (MINimum)

   No. Row name Activity Lower bound Upper bound
------ ------------ ------------- ------------- -------------
     1 ContrainteStock(C1,M1)
                               0 3
     2 ContrainteStock(C1,M2)
                               1 1
     3 ContrainteStock(C1,M3)
                               2 2
     4 ContrainteStock(C2,M1)
                               0 1
     5 ContrainteStock(C2,M2)
                               2 2
     6 ContrainteStock(C2,M3)
                               1 1
     7 ContrainteDemande(D1,C1)
```

```
                                 2 2 =
     8 ContrainteDemande(D1,C2)
                                 0 -0 =
     9 ContrainteDemande(D2,C1)
                                 1 1 =
    10 ContrainteDemande(D2,C2)
                                 3 3 =
    11 DefinitionBorneInfY(M1,D1,C1)
                                 0 -0
    12 DefinitionBorneInfY(M1,D1,C2)
                                 0 -0
    13 DefinitionBorneInfY(M1,D2,C1)
                                 0 -0
    14 DefinitionBorneInfY(M1,D2,C2)
                                 0 -0
    15 DefinitionBorneInfY(M2,D1,C1)
                                 0 -0
    16 DefinitionBorneInfY(M2,D1,C2)
                                 0 -0
    17 DefinitionBorneInfY(M2,D2,C1)
                                 0 -0
    18 DefinitionBorneInfY(M2,D2,C2)
                                 1 -0
    19 DefinitionBorneInfY(M3,D1,C1)
                                 1 -0
    20 DefinitionBorneInfY(M3,D1,C2)
                                 0 -0
    21 DefinitionBorneInfY(M3,D2,C1)
                                 0 -0
    22 DefinitionBorneInfY(M3,D2,C2)
                                 0 -0
    23 DefinitionBorneSup(M1,D1,C1)
                                 0 -0
    24 DefinitionBorneSup(M1,D1,C2)
                                 0 -0
    25 DefinitionBorneSup(M1,D2,C1)
                                 0 -0
    26 DefinitionBorneSup(M1,D2,C2)
                                 0 -0
    27 DefinitionBorneSup(M2,D1,C1)
                                 0 -0
    28 DefinitionBorneSup(M2,D1,C2)
                                 0 -0
    29 DefinitionBorneSup(M2,D2,C1)
                                -1 -0
    30 DefinitionBorneSup(M2,D2,C2)
                                -3 -0
    31 DefinitionBorneSup(M3,D1,C1)
                                -2 -0
    32 DefinitionBorneSup(M3,D1,C2)
                                 0 -0
    33 DefinitionBorneSup(M3,D2,C1)
                                 0 -0
    34 DefinitionBorneSup(M3,D2,C2)
                                -3 -0

   No. Column name Activity Lower bound Upper bound
------ ------------ ------------- ------------- -------------
     1 Y(M1,D1,C1) * 0 0 1
     2 Y(M1,D1,C2) * 0 0 1
     3 Y(M1,D2,C1) * 0 0 1
     4 Y(M1,D2,C2) * 0 0 1
     5 Y(M2,D1,C1) * 0 0 1
     6 Y(M2,D1,C2) * 0 0 1
     7 Y(M2,D2,C1) * 1 0 1
     8 Y(M2,D2,C2) * 1 0 1
     9 Y(M3,D1,C1) * 1 0 1
    10 Y(M3,D1,C2) * 0 0 1
    11 Y(M3,D2,C1) * 0 0 1
    12 Y(M3,D2,C2) * 1 0 1
    13 X(M1,D1,C1) * 0 0
```

```
    14 X(M1,D2,C1) * 0 0
    15 X(M2,D1,C1) * 0 0
    16 X(M2,D2,C1) * 1 0
    17 X(M3,D1,C1) * 2 0
    18 X(M3,D2,C1) * 0 0
    19 X(M1,D1,C2) * 0 0
    20 X(M1,D2,C2) * 0 0
    21 X(M2,D1,C2) * 0 0
    22 X(M2,D2,C2) * 2 0
    23 X(M3,D1,C2) * 0 0
    24 X(M3,D2,C2) * 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
```

Analyse des résultats :

L'analyse des résultats obtenus ci-dessus confirme que toutes les contraintes sont bien respectées. En effet, les contraintes **ContrainteStock** sont satisfaites, les quantités de colis sélectionnés pour répondre aux différentes demandes ne dépassent pas les stocks disponibles et correspondent exactement aux quantités demandées, comme validé par les contraintes **ContrainteDemande**.

Par ailleurs, les contraintes **DefinitionBorneInfY** et **DefinitionBorneSupY**, qui garantissent la définition correcte des variables booléennes y_{mdc}, sont également vérifiées.

En ce qui concerne les valeurs des différentes variables, elles sont cohérentes : les variables booléennes prennent uniquement les valeurs 0 ou 1, et les quantités sont toutes des nombres entiers, ce qui est logique puisque le nombre de colis est une grandeur discrète.

Enfin, le coût mimal calculé par notre programme, égal à de 435, semble cohérent avec les différents coûts (fixes et variables) présentés dans les tableaux de l'exemple. Cette valeur du coût minimal est également confirmée par un recalcul basé sur les valeurs des variables obtenues :

$$(20 + 90) + (20 + 90) + (5 + 100) + (10 + 100) = 435$$

Les fichiers fournis pour cette partie sont : `ModelECommerce2.mod.txt`, `DataECommerce2All.dat.txt`, `PbECommerce2.lp.txt` et `SolECommerce2.sol.txt`.

3.3 Cas particulier 2

Dans ce cas particulier, le livreur du magasin *ALPHA* quitte le magasin avec tous les colis pour les livrer à l'ensemble des clients. L'objectif est de minimiser la distance totale parcourue par le livreur. On peut d'ailleurs remarquer que ce problème est semblable au problème du **voyageur de commerce**.

3.3.1 Modélisation

Ainsi, la **fonction objectif** visera à minimiser la distance totale parcourue par le livreur, qui correspond à la somme des distances entre le magasin et le premier client, entre chaque client et celui qui suit, et enfin entre le dernier client et le magasin.

Les **variables** que nous avons considérées sont : les variables booléennes x_{ij} qui prennent la valeur vraie si le livreur se déplace de i vers j (où i et j peuvent être un client ou un magasin) et fausse dans le cas contraire. En outre, nous avons introduit les variables entières : $Ordre_i$ désignant l'ordre de passage du livreur par le lieu i.L'utilité de ces dernières sera détaillée dans la dernière contrainte mentionnée ci-dessous.

Les **contraintes** à respecter sont les suivantes : pour chaque client et le magasin, le livreur doit provenir d'un seul lieu, et en partant de n'importe quel lieu, le livreur ne peut se rendre qu'à un seul autre lieu. De plus, il est impossible pour le livreur de se déplacer d'un lieu vers lui-même, car cela serait illogique. Enfin, la dernière contrainte est liée à l'ordre de passage du livreur entre les différents lieux. En effet, si l'on ne prend pas en compte cet ordre, il est possible que plusieurs trajets différents soient proposés sans aucune relation entre eux, ce qui entraînerait une incohérence dans le parcours. Cela n'est pas souhaitable.
Pour modéliser cette contrainte, nous avons les relations suivantes :

— si $x_{ij} = 1$ alors : $Ordre_j = Ordrei + 1 \iff Ordre_i - Ordre_j = -1$

$$\iff x_{ij} + (Ordre_i - Ordre_j) \leq 0$$

— sinon ($x_{ij} = 0$) : $Ordre_i - Ordre_j \leq M$

Ainsi, en regroupant les deux cas, la contrainte devient :

$$x_{ij} \cdot (M+1) + (Ordre_i - Ordre_j) \leq M$$

où M représente une valeur suffisamment grande (le "big M"), qui sera choisie avec soin pour chaque instance du problème.

Voici la modélisation retenue pour notre problème.

Variables	**Fonction objectif**
x_{ij} : Variable booléenne indiquant si le livreur se déplace de i à j $Ordre_i$: Ordre dans lequel le livreur passe par i	min $\sum_{i,j} x_{ij} \cdot Distance_{ij}$
Contraintes	**Domaine**
$\sum_i x_{ij} = 1, \quad \forall j$ $\sum_j x_{ij} = 1, \quad \forall i$ $x_{ii} = 0, \quad \forall i$ $x_{ij} \cdot (M+1) + (Ordre_i - Ordre_j) \leq M, \quad \forall i,j$	$x_{ij} \in \{0,1\}, \quad \forall i,j$ $Ordre_i \in \{1,N\}, \quad \forall i$

Table 9 – Modélisation du problème de e-commerce pour le cas 2

3.3.2 Résultats obtenus

Afin de tester notre modèle, nous avons choisi de reprendre la matrice des distances donnée en TD et représentée ci-dessous :

	ALPHA	C1	C2	C3	C4	C5
ALPHA	-	1	10	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	-

Table 10 – Matrice des distances (magasin ALPHA et 5 clients à livrer).

```
Problem:
Rows: 48
Columns: 42 (42 integer, 36 binary)
Non-zeros: 158
Status: INTEGER OPTIMAL
Objective: DistanceTotal = 22 (MINimum)

   No. Row name Activity Lower bound Upper bound
------ ------------ ------------- ------------- -------------
     1 ContrainteColonnesX(ALPHA)
                                 1 1 =
     2 ContrainteColonnesX(C1)
                                 1 1 =
     3 ContrainteColonnesX(C2)
                                 1 1 =
     4 ContrainteColonnesX(C3)
                                 1 1 =
     5 ContrainteColonnesX(C4)
                                 1 1 =
     6 ContrainteColonnesX(C5)
                                 1 1 =
     7 ContrainteLignesX(ALPHA)
                                 1 1 =
     8 ContrainteLignesX(C1)
                                 1 1 =
     9 ContrainteLignesX(C2)
                                 1 1 =
    10 ContrainteLignesX(C3)
                                 1 1 =
    11 ContrainteLignesX(C4)
                                 1 1 =
    12 ContrainteLignesX(C5)
                                 1 1 =
    13 ContrainteX(ALPHA)
                                 0 -0 =
    14 ContrainteX(C1)
                                 0 -0 =
    15 ContrainteX(C2)
                                 0 -0 =
    16 ContrainteX(C3)
                                 0 -0 =
    17 ContrainteX(C4)
                                 0 -0 =
    18 ContrainteX(C5)
                                 0 -0 =
    19 ContrainteOrdre(ALPHA,C1)
                             10000 10000
    20 ContrainteOrdre(ALPHA,C2)
                                -5 10000
    21 ContrainteOrdre(ALPHA,C3)
                                -2 10000
    22 ContrainteOrdre(ALPHA,C4)
                                -3 10000
    23 ContrainteOrdre(ALPHA,C5)
                                -4 10000
    24 ContrainteOrdre(C1,C1)
                                 0 10000
    25 ContrainteOrdre(C1,C2)
                                -4 10000
    26 ContrainteOrdre(C1,C3)
                             10000 10000
    27 ContrainteOrdre(C1,C4)
                                -2 10000
    28 ContrainteOrdre(C1,C5)
                                -3 10000
    29 ContrainteOrdre(C2,C1)
                                 4 10000
    30 ContrainteOrdre(C2,C2)
                                 0 10000
    31 ContrainteOrdre(C2,C3)
                                 3 10000
```

```
    32 ContrainteOrdre(C2,C4)
                                 2 10000
    33 ContrainteOrdre(C2,C5)
                                 1 10000
    34 ContrainteOrdre(C3,C1)
                                 1 10000
    35 ContrainteOrdre(C3,C2)
                                -3 10000
    36 ContrainteOrdre(C3,C3)
                                 0 10000
    37 ContrainteOrdre(C3,C4)
                             10000 10000
    38 ContrainteOrdre(C3,C5)
                                -2 10000
    39 ContrainteOrdre(C4,C1)
                                 2 10000
    40 ContrainteOrdre(C4,C2)
                                -2 10000
    41 ContrainteOrdre(C4,C3)
                                 1 10000
    42 ContrainteOrdre(C4,C4)
                                 0 10000
    43 ContrainteOrdre(C4,C5)
                             10000 10000
    44 ContrainteOrdre(C5,C1)
                                 3 10000
    45 ContrainteOrdre(C5,C2)
                             10000 10000
    46 ContrainteOrdre(C5,C3)
                                 2 10000
    47 ContrainteOrdre(C5,C4)
                                 1 10000
    48 ContrainteOrdre(C5,C5)
                                 0 10000

   No. Column name Activity Lower bound Upper bound
------ ------------ ------------- ------------- -------------
     1 X(ALPHA,C1) * 1 0 1
     2 X(ALPHA,C2) * 0 0 1
     3 X(ALPHA,C3) * 0 0 1
     4 X(ALPHA,C4) * 0 0 1
     5 X(ALPHA,C5) * 0 0 1
     6 X(C1,ALPHA) * 0 0 1
     7 X(C1,C2) * 0 0 1
     8 X(C1,C3) * 1 0 1
     9 X(C1,C4) * 0 0 1
    10 X(C1,C5) * 0 0 1
    11 X(C2,ALPHA) * 1 0 1
    12 X(C2,C1) * 0 0 1
    13 X(C2,C3) * 0 0 1
    14 X(C2,C4) * 0 0 1
    15 X(C2,C5) * 0 0 1
    16 X(C3,ALPHA) * 0 0 1
    17 X(C3,C1) * 0 0 1
    18 X(C3,C2) * 0 0 1
    19 X(C3,C4) * 1 0 1
    20 X(C3,C5) * 0 0 1
    21 X(C4,ALPHA) * 0 0 1
    22 X(C4,C1) * 0 0 1
    23 X(C4,C2) * 0 0 1
    24 X(C4,C3) * 0 0 1
    25 X(C4,C5) * 1 0 1
    26 X(C5,ALPHA) * 0 0 1
    27 X(C5,C1) * 0 0 1
    28 X(C5,C2) * 1 0 1
    29 X(C5,C3) * 0 0 1
    30 X(C5,C4) * 0 0 1
    31 X(ALPHA,ALPHA)
                   * 0 0 1
    32 X(C1,C1) * 0 0 1
    33 X(C2,C2) * 0 0 1
```

```
    34 X(C3,C3) * 0 0 1
    35 X(C4,C4) * 0 0 1
    36 X(C5,C5) * 0 0 1
    37 Ordre(ALPHA) * 1 1 6
    38 Ordre(C1) * 2 1 6
    39 Ordre(C2) * 6 1 6
    40 Ordre(C3) * 3 1 6
    41 Ordre(C4) * 4 1 6
    42 Ordre(C5) * 5 1 6

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
```

Analyse des résultats :

L'analyse des résultats obtenus ci-dessus montre que toutes les contraintes sont bien respectées. En particulier, les contraintes **ContraintesColonnesX** et **ContraintesLignesX**, qui stipulent que chaque client et le magasin ainsi que le livreur doivent provenir d'un seul lieu, et qu'en partant de n'importe quel lieu, le livreur ne peut se rendre qu'à un autre seul lieu, sont toutes satisfaites (valeur égale à 1).

De plus, les contraintes **ContrainteOrdre**, associées à chaque lieu, sont également respectées : dans chaque cas, la valeur obtenue est inférieure ou égale à M (dans notre example, $M = 10000$).

Concernant les valeurs des variables définies, on constate que les variables booléennes ont bien le bon type (elles prennent uniquement les valeurs 0 ou 1), et que les variables entières représentant l'ordre de chaque lieu se situent bien entre 1 et 6 (6 étant le nombre de lieux dans cet exemple).

De plus, le chemin défini par les variables x_{ij} respecte les ordres défini par $Ordre_i$. En effet, le chemin proposé est le suivant :

$$ALPHA \rightarrow C1 \rightarrow C3 \rightarrow C4 \rightarrow C5 \rightarrow C2 \rightarrow ALPHA$$

En calculant la distance totale parcourue sur ce trajet, on obtient :

$$1 + 8 + 1 + 1 + 10 + 1 = 22$$

et ce résultat correspond bien à la distance affichée en haut du fichier obtenu correspondant à la distance totale parcourue par le livreur.

La valeur de la distance totale semble donc optimale, puisqu'elle est cohérente avec les différentes valeurs des distances entre les lieux. Ainsi, le fichier solution généré pour cette partie est jugé cohérent.

Les fichiers fournis pour cette partie sont : `ModelECommerceCasParticulier2.mod.txt`, `DataECommerceCasParticulier2.dat.txt`, `PbECommerceCasParticulier2.lp.txt` et `SolECommerceCasParticulier2.sol.txt`.