Recherche Opérationnelle :
Fascicule des TDs et TPs

Sandra U. Ngueveu, Corentin Boennec, Arthur Clavière, Aloïs Duguet, Théo Le Brun
ngueveu@laas.fr, boennec@laplace.univ-tlse.fr, arthur.claviere@onera.fr, {aduguet, tlebrun}@laas.fr

2022-2023

Table des matières

1 Préambule et consignes (A LIRE ATTENTIVEMENT)

1.1 Organisation des TDs et TPs

Trois sujets sont proposés. Ils seront traités lors des séances suivantes :

- **Sujet 1** : 1^{er} TD et 1^{er} TP (mi-novembre) ;
- **Sujet 2** : 2^{eme} TD et 2^{eme} TP (fin novembre/début décembre) ;
- **Sujet 3** : 3^{eme} TP (mi-décembre).

Travail en binôme : Chacun des trois sujets devra être réalisé en binôme. Etant donné le déroulement des TDs et TPs précisé ci-avant, il est bien évident que les deux membres d'un même binôme doivent appartenir au *même groupe de TP*. Des trinômes sont possibles, *à la marge*, notamment si le groupe de TP comporte un nombre impair d'élèves.

Conseil : pour les sujets traités sur deux séances, une bonne maîtrise du travail attendu lors de la première séance ne peut être que bénéfique pour la réalisation de la seconde séance (voir nécessaire).

1.2 Rendus

Chacun des trois sujets **de TP** proposés donnera lieu à un rendu *noté* (un rendu par binôme).
Dates de rendus : Les TP seront à rendre avant les dates suivantes (incluses) :

- TP1 25/11
- TP2 13/12
- TP3 06/01

Format : Pour chaque TP, le rendu **DOIT** être une *unique* archive, au format `.zip` ou`.tar.gz`, nommée :

`Sujet_{ID}_{NOM1}_{Prenom1}_{NOM2}_{Prenom2}`

où `ID` est le numéro du sujet, `NOM1` (respectivement `Prenom1`) est le nom du 1^{er} membre du binôme (respectivement le prénom du 1^{er} membre du binôme). Pour les noms ou prénoms composés, merci d'utiliser - (tiret du 6). Merci aussi de ne pas utiliser d'accent ou de caractère spécial dans les noms et prénoms des membres du binôme.
Par exemple, le rendu du sujet 2 par le binôme Pierre-Julien DUPONT et Léa DUBOIS DES-CHAMPS sera

`Sujet_2_DUPONT_Pierre-Julien_DUBOIS-DESCHAMPS_Lea.zip`
(ou `Sujet_2_DUPONT_Pierre-Julien_DUBOIS-DESCHAMPS_Lea.tar.gz`)

À propos du contenu de l'archive : Le contenu de l'archive dépendra des sujets :

- **Sujet 1** : un rapport au format `.pdf` ; les codes au format `.mod`, `.dat` et `.lp` ;
- **Sujet 2** : un jupyter notebook (incluant code + rapport) ; les instances du problème du sac à dos ;
- **Sujet 3** : un jupyter notebook (incluant code + rapport) ; les instances du problème du sac à dos.

Pour inclure de manière agréable à lire vos réponses aux questions dans un notebook, utiliser les cases Markdown disponibles sur jupyter notebook. Pour créer une cellule de type markdown, créer une cellule, puis sélectionner Cellule > Type de cellule > Markdown. Voici un tutoriel pour Markdown : https://documentation-snds.health-data-hub.fr/contribuer/guide_contribution/tutoriel_markdown.html

Envoi du rendu : Pour chaque sujet, l'archive doit être transmise grâce à l'emplacement dédié sur moodle par **UN SEUL** des deux membres du binôme.

1.3 Notation

Des barêmes indicatifs sont donnés dans chaque sujet. Ces barêmes précisent les points attribués à chaque question.

De manière générale, deux tiers de ces points correspondent à la réalisation *correcte* des éléments suivants :

- **Le code compile *sans erreur***. Dans le cas des jupyter notebooks, avant de rendre, pensez notamment à fermer le notebook et tester si tout fonctionne en le ré-ouvrant (de façon à s'assurer que tout fonctionne quand il est ouvert pour la première fois) ;
- **Le code implémente la ou les fonctions attendues** (résolution d'un problème donné, implémentation d'un algorithme donné) ;
- **Des tests et/ou expériences sont fournis et reproductibles** sans que l'utilisateur (en l'occurence le correcteur) n'ait à produire du code supplémentaire (appel à une fonction, affichage de la solution, etc).
- **Les éléments clés de l'implémentation sont expliqués** dans le rapport. Par exemple, une description des variables de décision utilisées pour le sujet 1, une description de la borne (ou des bornes) utilisée(s) pour le sujet 2 ;
- **Les résultats absurdes donnent lieu à un commentaire** et font réagir les membres du binômes. Typiquement, un probabilité de 4.10^3 (déjà vu) doit susciter un commentaire, tout comme une longueur égale à $+\infty$ quand il s'agit de trouver le plus court chemin dans un graphe (déjà vu aussi).
- **Les phrases du type "Le résultat est cohérent." sont étayées d'arguments.**

Le tiers restant correspond à la réalisation des éléments suivants :

- **Validation du code**, par exemple création d'une instance simple d'un problème donné, dont la solution est évidente, afin d'avoir un premier élément de validation du code ;
- **Qualité et lisibilité du code**, notamment les noms de variable qui doivent être explicites (`a`, `b`, `c` ne sont pas des noms de variables explicites), commentaires, indentation du code, etc ;
- **Analyse des paramètres d'un algorithme**, par exemple, pour le sujet 2, que se passe-t-il si on change de borne dans l'algorithme de BaB ? ;
- **Affichages intermédaires** lors de l'exécution du code, afin d'aider à la compréhension du code et de son exécution.

2 Sujet 1 : Modélisation+Résolution de PL/PLNE avec le solveur GLPK (TD+TP)

2.1 TD (1 séance)

Modéliser à la main les problèmes ci-après. (la résolution des modèles trouvés sera faite en TP et le barême par exercice correspond au rendu de TP).

2.1.1 Assemblage (3 points)

Dans une usine, une équipe d'ouvriers assemble deux modèles de voitures : le modèle L ("luxe"), à raison de 100 voitures en 6 heures, et le modèle S ("standard"), à raison de 100 voitures en 5 heures.

- Chaque semaine, l'équipe fournit au maximum 60 heures de travail.
- Toutes les voitures sont ensuite garées sur un parking qui est vidé chaque week-end, et dont la surface fait $15000m^2$. Une voiture L occupe $10m^2$, tandis qu'une voiture S occupe $20m^2$.
- De plus, il ne faut pas assembler plus de 800 voitures L par semaine, car la demande est limitée. En revanche, la demande en voitures S est tellement élevée qu'elle peut être considérée comme illimitée.
- Enfin, la marge (différence entre le prix de vente et le coût de production) vaut 10000€ pour une voiture L et 9000€ pour une voiture S.

L'usine souhaite savoir comment répartir le travail entre les deux modèles de voitures pour que la marge totale soit la plus grande possible.

Après avoir précisé un cas de figure où ce problème se modélise par PL et un cas de figure où ce problème se modélise par PLNE, proposer des modèles associés.

2.1.2 Gestion de personnel (4 points)

N personnes peuvent réaliser N travaux. Chaque personne doit être affectée à un travail et un seul. Chaque travail n'occupe qu'une seule personne. Toutes les personnes et tous les travaux doivent être affectés. Les personnes ont a priori, des compétences différentes pour la réalisation des travaux. Ceci se traduit par des coûts de formation $c(i, j)$ plus ou moins importants associés au couple personne i - travail j. On souhaite trouver l'affectation minimisant le coût total de formation. Etablissez un PLNE pour cela.

Suggestion : Si vous avez du mal à écrire le PLNE, focalisez vous sur un petit exemple de taille 3 : donnez-vous une matrice de coûts générés comme vous le souhaitez et tentez de formuler le problème qui aura pour but de trouver l'affectation minimisant le coût total en respectant les contraintes énoncées. Précisez les données choisies dans ce cas dans le rapport.

2.1.3 Applications en optimisation pour l'e-commerce

Parmi les problématiques d'optimisation émergeant en e-commerce, se trouvent l'affectation de commandes de clients aux magasins, compte tenu des coûts financiers et/ou environnementaux associés à la livraison des colis, à la préparation des commandes et à la gestion des différents stocks. Nous nous intéresserons particulièrement au problème d'affectation de commandes et tournées de véhicules pour différents magasins d'une même enseigne ou franchise à coût/impact minimal.

Cas particulier 1 (4 points)

Les tableaux (a), (b) et (c) représentent à titre d'exemple des demandes en fluide émanant de différentes commandes et les coûts unitaires associés selon les magasins d'origine. Chaque magasin dispose d'un volume de stock limité. Modéliser et résoudre à l'aide d'un programme linéaire.

	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

Cas particulier 2 (2 points)

En réalité, il ne s'agit pas de fluide mais de produits préconditionnés et une commande d'un client peut être constituée de plusieurs produits en quantités différentes. Le problème d'affectation consiste à déterminer le nombre de produits de chaque type livrés par chaque magasin à chaque client. Modifier et résoudre la formulation précédente pour tenir compte de la discrétisation de la demande.

Cas particulier 3 (4 points)

A présent nous souhaitons prendre en compte les coûts financiers et/ou environnementaux d'expédition des colis des magasins aux clients. Chaque magasin expédie, vers chaque client qu'il dessert, un unique colis contenant tous les produits fournis par ce magasin à ce client. Le coût résultant comprend un coût fixe (correspondant par exemple aux émissions polluantes du véhicule utilisé s'il était vide), et un coût variable dépendant de la quantité transportée (correspondant par exemple au surplus d'émissions dû à la charge transportée). Modifier la formulation précédente pour modéliser le problème résultant et résoudre avec les données des tableaux (d) et (e).

Indication : L'introduction d'une nouvelle variable de décision peut s'avérer utile ...

	M1	M2	M3
D1	110	90	100
D2	110	90	100

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

	M1	M2	M3
D1	10	1	5
D2	2	20	10

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

Cas particulier 4 (indépendant des autres) (3 points)

Un magasin ALPHA décide d'embaucher un livreur pour assurer toutes les livraisons qui le concerne. Ce dernier peut donc partir du magasin avec l'ensemble des colis et les livrer aux différents clients. (i) A quel problème théorique, vu en cours, ce cas particulier correspond-t-il ? (ii) Proposer un PLNE et l'utiliser pour résoudre l'instance associée à la matrice des distances suivantes du tableau (f). Préciser la solution obtenue.

	ALPHA	C1	C2	C3	C4	C5
ALPHA	-	1	1	10	12	12
C1	1	-	1	8	10	11
C2	1	1	-	8	11	10
C3	10	8	8	-	1	1
C4	12	10	11	1	-	1
C5	12	11	10	1	1	-

(f) matrice des distances (magasin ALPHA et 5 clients à livrer)

2.1.4 BONUS : Minimisation de trajets de livraison ou d'émissions polluantes

Minimisation des trajets de livraison

L'objectif est de modéliser un problème de minimisation des trajets de livraisons des commandes (i.e. distance parcourue) pour les livreurs des magasins. On fera donc abstraction des autres coûts pour se focaliser sur un problème mono-objectif. Le PLNE proposé doit prendre en entrée les données suivantes : les commandes à satisfaire (quantités et types de produits), les niveaux de stocks disponibles au sein des différents magasins, les temps de trajet entre les différents sites. A partir de ces données, le modèle doit pouvoir déterminer comment répartir les commandes entre les magasins et quelle est la tournée de livraison que doit réaliser le livreur de chaque magasin.

Données

— D : l'ensemble des commandes (équivalent à un ensemble clients car 1 client = 1 commande)
— P : l'ensemble des produits

— M : l'ensemble des magasins
— $N = M \cup D$: l'ensemble des noeuds (représentant les différents sites)
— $V = \{(i, j) \mid i, j \in N^2\}$ l'ensemble des arcs entre les noeuds.
— R : l'ensemble des $r_{ij} \in \mathbb{R}$: valeur de l'arc allant de i vers j représentant la distance à parcourir/temps de trajet entre les sites i et j
— Q : l'ensemble des $q_{dp} \in \mathbb{N}$: quantité de produit $p \in P$ dans la commande $d \in D$.
— S : l'ensemble des $s_{mp} \in \mathbb{N}$: stock de produit $p \in P$ dans le magasin $m \in M$.

Objectif

L'objectif est de minimiser la distance totale parcourue pour la livraison.

Contraintes

1. chaque magasin dispose de son propre livreur/camion qui sera en charge de livrer en une seule tournée tous les produits qui proviennent de son magasin (pas de limite de capacité sur les camions).
2. pour chaque magasin qui expédie au moins 1 produit, son livreur/camion débute sa tournée au magasin, visite une seule fois chacun des clients qu'il doit servir et retourne au magasin en fin de tournée.
3. chaque commande doit être satisfaite en totalité
4. une commande peut être satisfaite par un unique magasin qui livre tous les produits qui la composent, ou alors par plusieurs magasins, qui fournissent chacun une partie des produits
5. aucun magasin ne peut faire livrer plus de produits qu'il n'en possède en stock

Minimisation des émissions polluantes

A partir des mêmes données que précédemment, l'objectif est de modéliser le problème de minimisation du total des émissions polluantes (et non plus la distance parcourue), pour la livraison.

Données

En plus de données déjà décrites précédemment dans la partie minimisation des trajets de livraison, connaissant les caractéristiques des véhicules de transport choisis et vitesses de circulation, nous pouvons exprimer à l'aide de l'équation (1) la quantité d'émissions polluantes d'un véhicule transportant une quantité totale de produits q sur une distance r.

$$EP(q, r) = Aqr + Br + C \tag{1}$$

Des valeurs des constantes pourraient être $A = 1,18.10^{-5}$, $B = 2,14.10^{-1}$ et $C = 0$

Objectif

Minimiser le total des émissions polluantes.

Contraintes

Identiques aux contraintes énoncées précédemment dans la partie de minimisation des trajets de livraison.

2.2 TP (1 séance)

2.2.1 Implémentation et résolution avec GLPK

Après avoir suivi le tutoriel GLPK de la section 2.2.2, implémenter et résoudre avec GLPK (en format lp et/ou format gmpl pour une solution plus générique) les modèles que vous avez précédemment proposés pour les problèmes de la section 2.1.

1. Expliciter dans le rapport votre choix de variable de décision et leur domaine de définition
2. Justifier la pertinence du format utilisé selon le problème
3. Faire apparaître les résultats obtenus dans le rapport et dire si cela vous paraît cohérent. Pourquoi ?

2.2.2 Bref tutoriel Solveur Executable GLPSOL

Le solveur glpsol de GLPK est un exécutable (avec sa .dll si windows, sans sinon) permettant de lire et résoudre les problèmes formulés soit sous format ".lp", soit sous format GMPL (dérivé de l'AMPL). Il se manipule en ligne de commande tel que suit :

>> ./glpsol [options] [nom_des_fichiers]

La liste des instructions disponibles s'obtient avec la commande :

>> ./glpsol −−help

Pour débuter, exécuter le script "./InstructionsGLPK.txt". Il se chargera de télécharger le code source de GLPK et le compiler pour générer l'exécutable.

Format ".lp"

PbCiment.lp.txt

```
\* Problem: Fabrication de ciment *\

Maximize
 Benefice: + 50 Q1 + 70 Q2

Subject To
 CapaciteduFour: + 40 Q1 + 12 Q2 <= 360
 CapacitedeBroyage: + 20 Q1 + 30 Q2 <= 480

End
```

FIGURE 4 – Exemple du fichier "PbCiment.lp.txt"

Lire et résoudre un problème PbCiment.lp.txt écrit sous format ".lp" :

>> ./glpsol −−lp PbCiment.lp.txt

— écrire la solution dans un fichier (version lisible)

>> ./glpsol −−lp PbCiment.lp.txt −o SolCiment.sol.txt

— écrire la solution dans un fichier (version chargeable)

>> ./glpsol −−lp PbCiment.lp.txt −w SolCiment.sol.txt

Format gmpl

Lire et résoudre un problème posé sous format GMPL :

— la section “modèle" et la section “données" (optionnelle) sont dans le même fichier :
>> ./glpsol −m ModelCimentwithData.mod.txt

— la section “données" est dans un autre fichier (ignorer celle du fichier modèle) :
>> ./glpsol −m ModelCiment.mod.txt −d DataCimentAll.dat.txt
>> ./glpsol −m ModelCimentwithData.mod.txt −d DataCimentAll.dat.txt

ModelCiment.mod.txt

```
#  GLPK model file created by SUN for RO teaching

############################  Model ############################

############################  Sets  ############################

set CIMENTS;

set MACHINES;

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

var Q{i in CIMENTS}, >=0;
# or binary ?;

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

param benefparciment{i in CIMENTS};

param capacite{i in MACHINES};

param dureefab{i in MACHINES, j in CIMENTS};

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


s.t. RespectCapaciteDeChaqueMachine{j in MACHINES}:
  sum{i in CIMENTS} dureefab[j,i] * Q[i] = capacite[j];


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

maximize BeneficeTotal:
                sum{i in CIMENTS} Q[i]* benefparciment[i];


end;
```

DataCimentAll.dat...

```
# Autres données possibles pour
# le problème de fabrication
# de ciment

data;

set CIMENTS :=
11
22;

set MACHINES :=
Four2
Broyeuse2;

param dureefab: 11 22 :=
Four2 40 12
Broyeuse2 20 30;

param capacite :=
Four2 360
Broyeuse2 480;

param benefparciment :=
11 500
22 700;

end;
```

FIGURE 5 – Exemples de fichiers “ModelCiment.mod" et “DataCiment.dat"

— générer le fichier “*.lp" à partir du modèle GMPL
>> ./glpsol −m ModelCiment.mod.txt −d DataCimentAll.dat.txt −−wlp PbCiment2.lp.txt

— vérifier (sans résoudre) que la syntaxe du problème est correcte
>> ./glpsol −−check −m ModelCiment.mod.txt −d DataCimentAll.dat.txt

D’autres exemples et illustrations sont disponibles à l’adresse :
`http://connect.ed-diamond.com/GNU-Linux-Magazine/GLMF-135/Decouverte-du-solveur-GLPK`

3 Sujet 2 : Algorithme de Branch-and-Bound (TD+TP)

3.1 TD (1 séance)

3.1.1 Composants (déjà vu en cours)

1. Ecrire un modèle ($\mathcal{P}$) de Programmation Linéaire en Nombres Entiers (PLNE) du problème de sac à dos avec n objets, de valeurs $c_i > 0$ et de poids $w_i > 0$ pour $i = 1, ..., n$, et pour une capacité C.
2. Quels sont les points respectant uniquement les contraintes d'intégralité ? Combien y en a-t-il ? Pourquoi ne pas énumérer tous les points respectant ces contraintes pour résoudre le problème ?
3. Donner une solution entière non réalisable, et une solution entière réalisable (on suppose $0 < C < \sum_{i=1}^{n} w_i$). En déduire une borne inférieure et une borne supérieure du problème ($\mathcal{P}$).
4. Ecrire la relaxation linéaire du modèle de PLNE, appelée ($\mathcal{RLP}$). Pourquoi est-ce que c'est une relaxation du modèle de PLNE ?
5. Donner une méthode n'utilisant pas la programmation linéaire pour obtenir une borne supérieure de ($\mathcal{P}$) et montrer pourquoi cela calcule une borne supérieure.

3.1.2 Résolution d'une instance du problème de sac à dos

L'objectif est de résoudre ($\mathcal{P}$) à l'aide d'un algorithme de Branch-and-Bound. Prendre pour cela comme paramètres la règle de séparation, les tests de sondabilité et la stratégie d'exploration de **la page 13/24 du cours**. Prendre comme calcul de borne supérieure celle décrite à la question 5 de l'exercice 3.1.1.

Instance du sac à dos :

Les paramètres de l'instance du problème de sac à dos de cet exercice sont donnés dans le tableau (6) ci-dessous. c_i est la valeur de l'objet i, et w_i le poids de l'objet i. Définir la relaxation linéaire de cette instance comme problème du noeud 0 de l'arborescence du Branch-and-Bound. On appelle

	1	2	3	4
c_i	42	40	12	25
w_i	7	4	3	5

FIGURE 6 – Instance pour TD sac à dos, avec capacité du sac $C = 10$

capacité restante la capacité du problème de sac à dos du noeud concerné.

Questions :

1. Faire passer le test d'admissibilité au noeud 0.
2. Quelle est la solution d'un problème de sac à dos de capacité strictement négative ?
3. Faire passer le test d'optimalité au noeud 0.
4. Est-ce que la borne supérieure trouvée est plus grande que la borne inférieure décrite à la question 3 de l'exercice 3.1.1 ? Expliquer pourquoi c'était attendu.
5. Faire passer le test de résolution au noeud 0.
6. Quand un test de résolution renvoie vrai, qu'est-ce que cela indique pour la solution optimale du noeud en question ?
7. Conclure quant à la sondabilité du noeud 0.
8. Trouver k l'indice de variable à choisir selon la règle de séparation appliquée au noeud 0.
9. Ecrire le problème du noeud 1 obtenu en ajoutant la contrainte "$x_k = 1$" au problème du noeud 0. Faire de même pour le noeud 2 avec la contrainte "$x_k = 0$". Quel est le lien entre les points admissibles intégraux du noeud 0 et l'union des points admissibles intégraux des noeuds 1 et 2 ?
10. Ecrire les modèles mathématiques correspondants aux problèmes résultants aux noeuds 1 et 2. Constater qu'il s'agit d'instances de sac à dos avec moins d'objets et une nouvelle capacité, appelée capacité restante, en référence au problème $(\mathcal{P})$.
11. Suivre le diagramme page 11/24 du cours pour finir l'exécution de l'algorithme du Branch-and-Bound. Dessiner en même temps une arborescence des noeuds comme vu en cours après la page 22/24 : faire figurer les informations importantes de cette arborescence comme la valeur de la borne supérieure, un test qui retourne vrai, la contrainte ajoutée lors de la création d'un noeud...
12. Quelle est la solution optimale de cette instance de sac à dos ? Quelle est la valeur optimale associée ? Donner le ratio nombre de noeuds exploré sur nombre de solution respectant uniquement les contraintes d'intégralités ?

3.1.3 Branch-and-Bound classique pour PLNE

Soit le problème de PLNE suivant :

$$(\mathcal{P}') \quad \begin{cases} Max & x + y \\ s.c. & 3x + y \leq \frac{21}{2} \\ & y \leq \frac{5}{2} \\ & x, y \in \{0, 1\} \end{cases}$$

Les paramètres de Branch-and-Bound utilisés dans cet exercice sont ceux décrits slide 19/24. Définir la relaxation linéaire de $(\mathcal{P}')$ comme problème du noeud 0 de l'arborescence du Branch-and-Bound.

1. Représenter graphiquement l'ensemble admissible du noeud 0. Faire de même pour l'ensemble admissible de $(\mathcal{P}')$. Tracer le vecteur-gradient correspondant à la fonction-objectif.
2. Faire passer les tests de sondabilité au noeud 0. La borne supérieure se trouve grâce à une résolution graphique du simplexe.
3. Appliquer la règle de séparation au noeud 0.
4. Représenter visuellement l'ensemble admissible des noeuds 1 et 2. Comment est leur union par rapport à l'ensemble admissible du noeud 0 ?
5. Représenter visuellement l'ensemble des points entiers des problèmes des noeuds 1 et 2. Comment est leur union par rapport à l'ensemble admissible du problème $(\mathcal{P}')$?
6. Finir l'exécution de l'algorithme du Branch-and-Bound. Donner la solution du problème $(\mathcal{P}')$ ainsi que la valeur de la fonction-objectif associée.

3.2 TP (1 séance)

Le but de ce TP est de résoudre différentes instances du problème du sac à dos à l'aide d'un algorithme de branch-and-bound que vous aurez implémenté.
Rendus attendus (à soumettre sur moodle à l'emplacement dédié) : archive .zip contenant :
- le notebook en julia permettant de résoudre les instances fournies et d'autres de votre choix
- les instances lancées dans le notebook
- les réponses aux questions, les justifications de programmation et les commentaires du code inclus dans le notebook

3.2.1 Prise en main

Télécharger de Moodle le zip contenant :
- un dossier instancesETU contenant les jeux de données disponibles pour vos évaluations numériques (vous pouvez aussi en ajouter d'autres pour illustrer des cas particuliers)
- un notebook-exemple fourni, qui ne fonctionne que pour l'instance test.opb.txt correspondant à l'exemple du cours, et utilise un solveur de LP nommé Clp pour résoudre les relaxations linéaires

3.2.2 Travail à faire

Questions préliminaires (4 points) :
Dans le Branch-and-Bound fourni dans le notebook-exemple :

1. Quelle est la règle de séparation ?

2. Quelle est la méthode de calcul de borne supérieure ?
3. Quels sont les tests de sondabilité TA, TO, TR ?
4. Quelle est la stratégie d'exploration ?

Code et analyse (16 points) :

1. Coder les calculs de bornes 1 et 2 décrits page 6/24 du cours (la borne 2 à coder est la version utilisant un tri des objets par ordre décroissant de r_i). Une option doit permettre de choisir laquelle de ces deux bornes utiliser.
2. Adapter l'algorithme de Branch-and-Bound pour fonctionner sans appel à JuMP ou à un solveur de programmation linéaire. Intégrer les calculs de bornes implémentés à la question précédente comme méthode d'obtention de borne supérieure. Un argument de la fonction principale doit permettre de sélectionner l'instance du dossier instancesETU à résoudre. Attention : supprimer la section dédiée à Clp supprimera non seulement le modèle mathématique model2, mais aussi le vecteur des variables x, et la liste des variables binaires par ordre de priorité pour branchement varsshouldbebinary. A vous de corriger le code complet en conséquence pour que votre algorithme fonctionne.
3. Donner les points clés de votre implémentation des différents blocs du Branch-and-Bound (règle de séparation, TA, TO, TR, stratégie d'exploration).
4. Expliquer votre choix de structure de données permettant de garder les informations nécessaires au Branch-and-Bound (ce qui veut dire faire fonctionner les fonctions pour la règle de séparation, TA, TO, TR, et la stratégie d'exploration).
5. Comparer les performances du Branch-and-Bound en utilisant la borne 1 ou la borne 2. Le résultat vous semble-t-il cohérent ? Argumenter. Remarque : la comparaison doit utiliser suffisamment d'instances et de tailles variées.

3.2.3 BONUS

Comparer votre Branch-and-Bound avec GLPK : une comparaison en temps ne serait pas forcément pertinente au vue des différences de langages de programmation, mais une comparaison avec le nombre de noeuds de l'arborescence serait intéressante à faire, en particulier avec différents paramètres de choix de stratégies d'exploration de GLPK

4 Sujet 3 : Programmation dynamique (TP)

4.1 TP (1 séance)

4.1.1 Résolution du problème du sac à dos

Le but est de résoudre le problème du sac à dos et le problème du plus court chemin dans un graphe à l'aide d'un algorithme de programmation dynamique que vous aurez implémenté.
Rendus attendus (à soumettre sur moodle à l'emplacement dédié) : archive .zip contenant :

- le notebook en julia permettant de résoudre les instances fournies et d'autres de votre choix
- les instances lancées dans le notebook
- les analyses demandées et les commentaires du code (inclus dans le notebook)

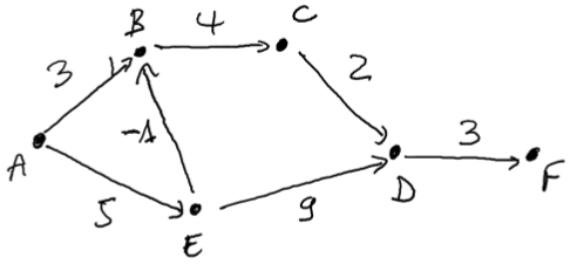
4.1.2 Travail à faire

- Implémenter en Julia l'algorithme de programmation dynamique permettant de résoudre le problème du sac à dos.
- Donner une courte argumentation de l'adéquation du résultat avec l'instance résolue
- Expliquer le fonctionnement de votre algorithme
- Tester l'algorithme sur plusieurs instances de tailles différentes fournies dans le TP2, donner les solutions obtenues et les valeurs de fonction-objectif. Comparer avec celles obtenues avec le branch-and-bound implémenté lors des TP2.

4.1.3 BONUS : Algorithme de Bellman-Ford

Implémenter en Julia l'algorithme de Bellman-Ford permettant de calculer le plus court chemin entre un sommet source s et tous les autres sommets d'un graphe quelconque. A titre d'exemple vous pouvez vous baser sur l'exemple rappelé ci-après. Tester ensuite avec différents jeux de données de votre choix.

Exemple



itération	A	B	C	D	E	F
0	0	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$
1	0	3^A	$+\infty$	$+\infty$	5^A	$+\infty$
2	0	3^A	7^B	14^E	5^A	$+\infty$
3	0	3^A	7^B	9^C	5^A	17^D
4	0	3^A	7^B	9^C	5^A	12^D
5	0	3^A	7^B	9^C	5^A	12^D

FIGURE : Exemple : graphe et tableau des valeurs de f_i^k obtenu lors de l'application de l'algorithme de Bellman-Ford pour calculer le plus court chemin entre les noeuds A et F du graphe fourni

Données

- Ce graphe a $n = 6$ sommets (A, B, ..., F) et $m = 7$ arcs ((AB), (AE), ... (DF))
- Chaque arc ij a un coût c_{ij} (par exemple $c_{AB} = 3$)

Relation de récurrence

- Soit f_i^k la valeur du plus court chemin du sommet de départ (A) et un sommet i calculé à l'itération k
- La relation de récurrence qui s'applique est $f_i^k = \min_{\forall j \in \text{Pred}(i)} f_j^{k-1} + c_{ji}$ avec $i \in \{1, ..., n\}$ et $k \geq 1$

Condition d'arrêt

- L'algorithme de Bellman-Ford s'arrête dès l'itération k qui vérifie soit $\forall i \ f_i^k = f_i^{k-1}$, soit $k >= n + 1$.
- Remarque : Si le cas de figure $k = n + 1$ se produit, alors cela démontre l'existence d'un cycle de longueur négative dans le graphe. Selon ce que modélise ce graphe, il se peut que cela permette de détecter une incohérence.

Solution obtenue

- Le plus court chemin entre A et F est A-B-C-D-F. Ce chemin a un coût de 12.