

Recherche Opérationnelle :

Fascicule des TDs et TPs

Sandra U. Ngueveu, Louis Bonnet, Noémie Cohen, Camille Lescuyer, Théo Le Brun
ngueveu@laas.fr, camille.lescuyer@airbus.com, noemie-lucie-virginie.cohen@airbus.com,
{lbonnet, tlebrun}@laas.fr

2023-2024

Table des matières

1	Préambule et consignes (A LIRE ATTENTIVEMENT)	2
1.1	Organisation des TDs et TPs	2
1.2	Rendus	2
1.3	Notation	3
2	Sujet 1 : Modélisation+Résolution de PL/PLNE avec le solveur GLPK (TD+TP)	5
2.1	TD (1 séance)	5
2.2	Assemblage (3 points)	5
2.3	Affectation avec prise en compte des préférences (4 points)	5
3	Applications en optimisation pour l'e-commerce	6
3.1	Cas particulier 1.1 (5 points)	6
3.2	Cas particulier 1.2 (3 points)	7
3.3	Cas particulier 2 (3points)	7
3.4	Minimisation des émissions polluantes (2 points + 2 points bonus)	7
3.5	TP (1 séance)	10
4	Sujet 2 : Algorithme de Branch-and-Bound (TD+TP)	12
4.1	TD (1 séance)	12
4.2	TP (1 séance)	14
5	Sujet 3 : Programmation dynamique (TP)	17
5.1	TP (1 séance)	17

1 Préambule et consignes (A LIRE ATTENTIVEMENT)

1.1 Organisation des TDs et TP

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 TP 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 28/11
- TP2 22/12
- TP3 12/01

1 point de pénalité sera appliqué par journée de retard.

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 DESCHAMPS 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` ; les fichiers solutions au format `txt` ;
- **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 barèmes indicatifs sont donnés dans chaque sujet. Ces barèmes 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'occurrence 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 intermédiaires** 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 barème par exercice correspond au rendu de TP).

2.2 Assemblage (3 points)

Dans une usine, une équipe d'ouvriers assemble des vélos cargos et des vélos standards : les vélos cargos (C), à raison de 100 modèles en 6 heures, et les vélos standards (S), à raison de 100 modèles en 5 heures.

- Chaque semaine, l'équipe fournit au maximum 60 heures de travail.
- Tous les véhicules sont ensuite garés sur un parking qui est vidé chaque week-end, et dont la surface fait $1500m^2$. Un vélo cargo C occupe $2.5m^2$, tandis qu'un vélo standard S occupe $1m^2$.
- De plus, il ne faut pas assembler plus de 700 vélos cargos C par semaine, car les ressources nécessaires aux batteries sont limitées. En revanche, les vélos standards S, ne sont pas limités par l'approvisionnement en ressources.
- Enfin, la marge (différence entre le prix de vente et le coût de production) vaut 700€ pour un vélo cargo et 300€ pour un vélo standard S.

L'usine souhaite savoir comment répartir le travail entre les deux modèles de vélos 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.

2.3 Affectation avec prise en compte des préférences (4 points)

Une manageuse doit préparer le planning de son 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, comme illustré en figure 1. Ainsi, $c(i, j)$ correspond au score de préférence de la personne P_i pour la tâche T_j . La manageuse souhaite trouver la meilleure affectation possible. 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 scores de préférence comme vous le souhaitez et tentez de formuler

le problème qui aura pour but de trouver la meilleure affectation tout en respectant les contraintes énoncées.

	T_1	T_2	T_3	...	T_N
P_1	4	7	9
P_2	9	8	3
P_3	2	1	2
...	1	3	1
	2	2	4
P_N	4	3	2

FIGURE 1 – Exemple de matrice de préférences

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. Ces affectations dépendent 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.

3.1 Cas particulier 1.1 (5 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 (dans le cas de livraison de fluides) ou programme linéaire en nombres entiers (dans le cas de livraison de colis).

	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

3.2 Cas particulier 1.2 (3 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).

	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

3.3 Cas particulier 2 (3points)

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)

3.4 Minimisation des émissions polluantes (2 points + 2 points bonus)

L'objectif est de modéliser un problème de minimisation des émissions polluantes des trajets de livraison des commandes par 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, les caractéristiques des véhicules. 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.

3.4.1 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 $m \in M$.

De plus, connaissant les caractéristiques des véhicules de transport choisis et les 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 .

$$EP(q, r) = Aqr + Br + C \quad (1)$$

avec $A = 1,18.10^{-5}$, $B = 2,14.10^{-1}$ et $C = 0$

3.4.2 Objectif

L'objectif est de minimiser le total des émissions polluantes.

3.4.3 Contraintes

- 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).
- 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.
- chaque commande doit être satisfaite en totalité
- 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

— aucun magasin ne peut faire livrer plus de produits qu'il n'en possède en stock

MENTION SPECIALE

Données assainies et cahier des charges tirés du stage de Théo Le Brun et de la chaire de formation et de recherche "Retail Responsable" 2022-2027 de Sandra Ulrich Ngueveu.



<https://www.laas.fr/projects/ChaireRetailResp/>

3.5 TP (1 séance)

3.5.1 Implémentation et résolution avec GLPK

Après avoir suivi le tutoriel GLPK de la section 3.5.2, implémenter et résoudre avec GLPK (en format lp 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 variables 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 ?

3.5.2 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 débiter, exécuter le script “./InstructionsGLPK.txt”. Il se chargera de télécharger le code source de GLPK et le compiler pour générer l’exécutable.

Format “.lp”

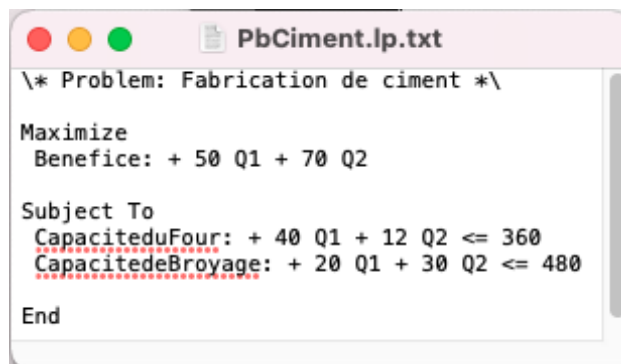


FIGURE 5 – Exemple du fichier “PbCiment.lp.txt”

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
```

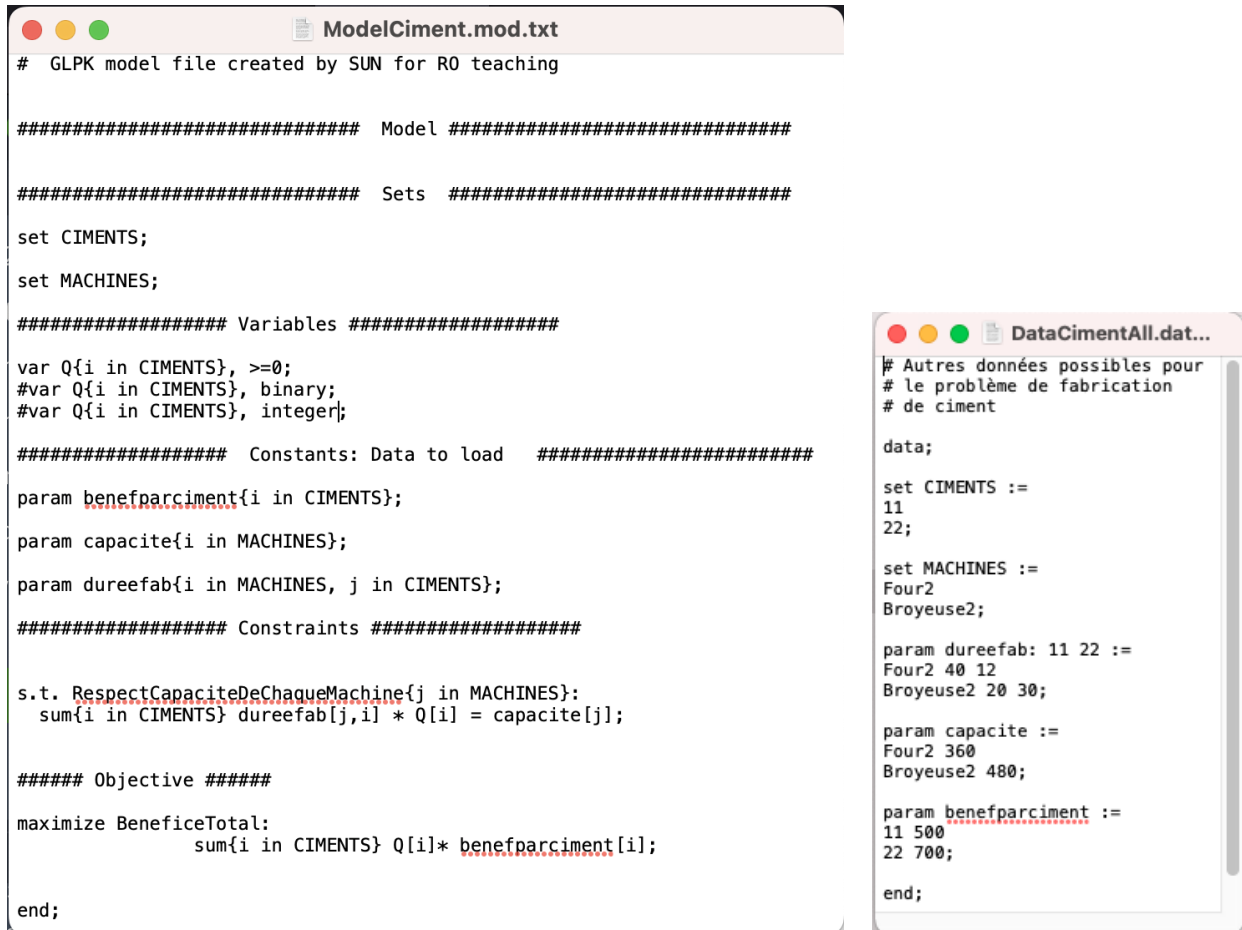


FIGURE 6 – Exemples de fichiers “ModelCiment.mod” et “DataCiment.dat”

— 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>

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

4.1 TD (1 séance)

4.1.1 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 $i = 1, \dots, n$, et pour une capacité C .
2. Les contraintes d'intégrité sont les contraintes définissant les variables non continues, comme $x_1 \in \mathbb{N}$ et $x_1 \in \{0, 1\}$. Quels sont alors les points respectant uniquement les contraintes d'intégrité? 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? Montrer pourquoi une solution obtenue via cette relaxation linéaire est une borne supérieure de notre problème.
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.

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

L'objectif de cet exercice est de reprendre l'exemple vu en cours concernant la résolution du sac à dos (\mathcal{P}) avec l'algorithme du Branch-and-Bound mais avec des règles différentes (TA, TO, TR, règle de séparation et stratégie d'exploration) dans le but d'observer les effets de ces nouveaux choix de règles sur l'arborescence du Branch-and-Bound.

Méthode de calcul de borne supérieure (= résolution de la relaxation) : Trier les objets par ratio décroissant (ratio $r_i = \frac{c_i}{w_i}$). Sélectionner en entier les objets disponibles de plus grand ratio tel que la capacité du sac n'est pas dépassée. Dès qu'un objet ne peut plus rentrer entièrement dans le sac, n'en prendre que la fraction permettant de remplir le sac.

Règle de séparation : Brancher sur la variable la plus fractionnaire (i.e. dont la partie décimale est la plus proche de 0.5).

Règle d'exploration : Prioriser le noeud ayant la plus grande borne supérieure.

Tests de sondabilité : Test d'admissibilité (TA) réussit si que la capacité restante est négative. Test d'optimalité (TO) réussit si que la borne supérieure obtenue est moins bonne que la meilleure solution connue. Test d'intégrité (TI) réussit si le calcul de la borne supérieure donne des valeurs entières aux variables.

Instance du sac à dos :

Les paramètres de l'instance du problème de sac à dos de cet exercice sont donnés dans le tableau (7) 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 7 – 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, en décrivant le calcul de borne supérieure.
4. Est-ce que la borne supérieure trouvée est plus grande que la borne inférieure décrite à la question 3 de l'exercice 4.1.1 ? 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. Simplifier les modèles mathématiques des 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. Finir l'exécution du Branch and Bound en dessinant l'arborescence et en y faisant figurer les informations importantes 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'intégrités ?

4.1.3 Branch-and-Bound classique pour PLNE

Soit le problème de PLNE suivant :

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

Les paramètres de Branch-and-Bound utilisés dans cet exercice sont les mêmes que ceux décrits dans la partie précédente, à l'exception de la méthode de calcul de la borne supérieure qui se résout par la méthode graphique. 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.

4.2 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, **ET UNIQUEMENT CELLES UTILISÉES**. Des points seront retirés le cas échéant.
- les réponses aux questions, les justifications de programmation et les commentaires du code inclus dans le notebook

4.2.1 Prise en main

Télécharger de Moodle le zip contenant :

- un dossier InstancesKnapSack 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 fonctionne pour l'instance test.opb.txt correspondant à l'exemple du cours. Le notebook utilise le package JuMP pour la modélisation de la relaxation linéaire et un solveur de LP nommé Clp pour sa résolution

4.2.2 Travail à faire

Questions préliminaires (4 points) :

Dans le Branch-and-Bound fourni dans le notebook-exemple (pas dans le code que vous avez modifié) :

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

Code et analyse (16 points) :

1. Coder le calcul de la borne 1 vue en cours, consistant à combler la capacité restante du sac à partir de l'objet de meilleur ratio.
2. Adapter l'algorithme de Branch-and-Bound pour fonctionner sans appel à JuMP ou à Clp. 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. Coder le calcul de la borne 2 décrit précédemment en section 4.1.2. Une option de la fonction principale doit permettre de choisir laquelle des bornes 1 et 2 utiliser.
4. Donner les points clés de votre implémentation des différents blocs du Branch-and-Bound (borne supérieure, règle de séparation, TA, TO, TR, stratégie d'exploration) et expliquer votre choix de structure de données permettant de garder les informations nécessaires au Branch-and-Bound.
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.

6. Proposer des solutions pour améliorer le Branch and Bound (modifications des règles de séparation, modification des règles d'exploration, résolution de la relaxation...). Bonus : implémenter et faire des tests comparatifs si possible.

5 Sujet 3 : Programmation dynamique (TP)

5.1 TP (1 séance)

5.1.1 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 :

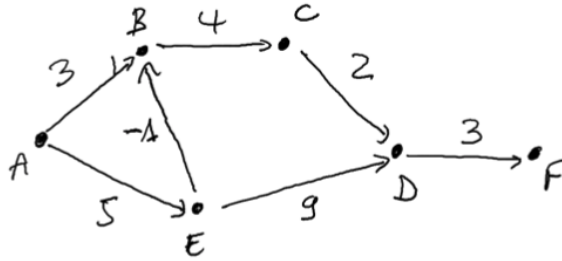
- le notebook en julia permettant de résoudre les instances fournies et d'autres de votre choix
- les instances lancées dans le notebook
- les analyses demandées et les commentaires du code (inclus dans le notebook)

5.1.2 Travail à faire (barème indicatif)

- Expliquer ce qu'est la programmation dynamique et la relation de récurrence associée à un problème de programmation dynamique (1 point)
- Implémenter en Julia l'algorithme de programmation dynamique permettant de résoudre le problème du sac à dos (9 points)
- Donner une courte argumentation de l'adéquation du résultat avec l'instance résolue (1 point)
- Expliquer le fonctionnement de votre algorithme (3 points)
- Tester l'algorithme sur plusieurs instances de tailles différentes fournies dans le TP2, donner les solutions obtenues et les valeurs de fonction-objectif (2 points)
- En utilisant seulement des instances à au moins 20 objets, comparer les temps de calcul obtenus avec votre algorithme de programmation dynamique et votre branch-and-bound implémenté lors du TP2 (ou à défaut la version fournie avec la borne calculée via JuMP/relaxation linéaire). Observer les éventuelles différences. Comment expliquer ces observations ? (4 points)

5.1.3 BONUS : Algorithme de Bellman-Ford (3 points)

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 obtenues 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 $f_i^k = \min_{j \in \text{Pred}(i)} f_j^{k-1} + c_{ji}$ avec $i \in \{1, \dots, n\}$ 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 $k \geq n + 1$.
- 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.