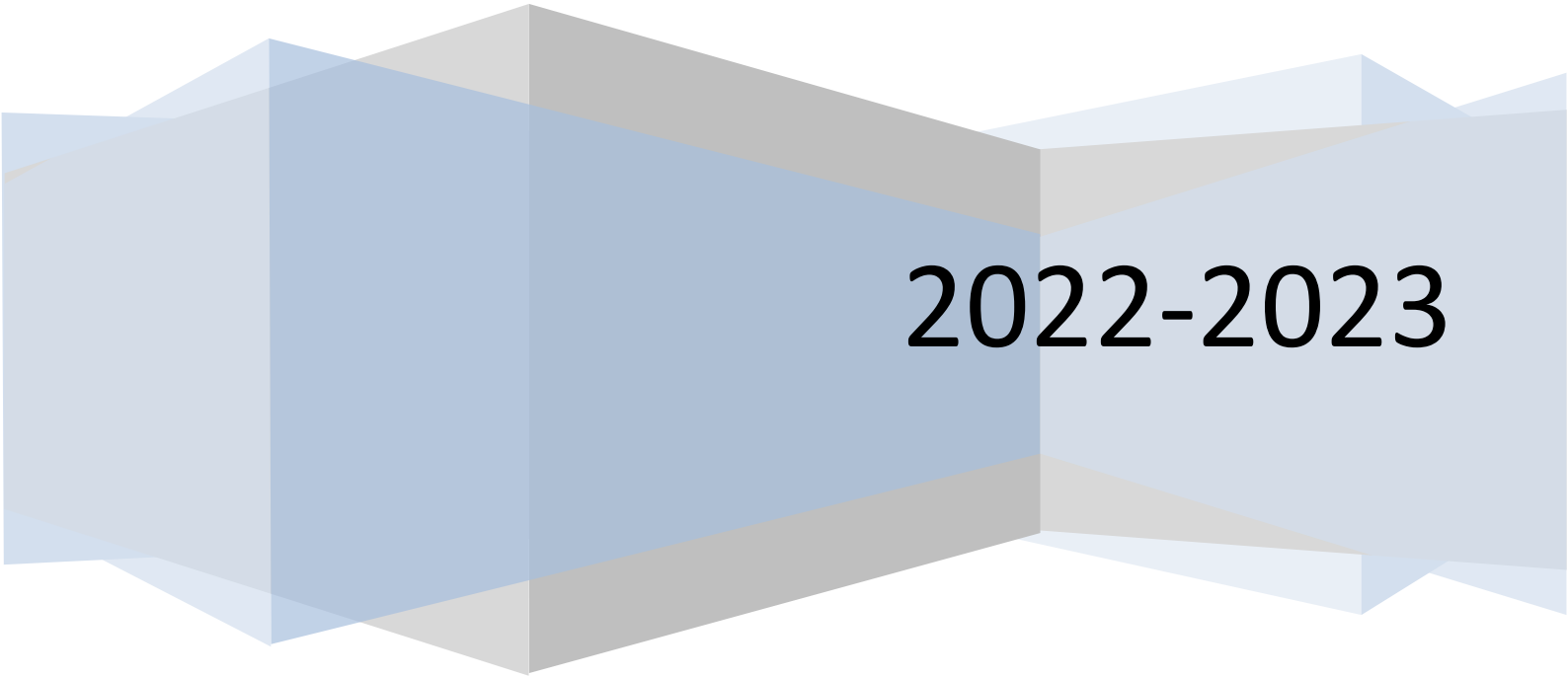


Eric Fusy (initial 1)

Compte rendu du TP1

Optimisation

LY-IENG Steven / DOS SANTOS Loïc



2022-2023

Sommaire

| | |
|------------------------------------|---|
| 1. Prise en main de lp solve | 3 |
| Exercice 1 | 3 |
| Exercice 2 | 3 |
| Exercice 3 | 3 |
| 2. Un problème générique | 3 |
| Exercice 4 | 3 |
| Exercice 5 | 4 |
| Exercice 6 | 4 |
| Exercice 7 | 4 |
| Exercice 8 | 5 |
| 3. Un problème de découpe | 5 |
| Exercice 9 | 5 |
| Exercice 10 | 5 |
| Exercice 11 | 6 |

1. Prise en main de lp solve

Exercice 1

Vérifiez que tout fonctionne correctement dans l'exemple ci-dessus.

Quelles sont les valeurs optimales de x_1 et x_2 si on se restreint à des nombres entiers ?

Si on se restreint aux entiers les valeurs optimales sont :

- $x_1 = 2$
- $x_2 = 8$

Exercice 2

lp solve impose automatiquement des contraintes de non-négativité pour toutes les variables.

Essayez de changer l'objectif de l'exemple à max: $-2x_1 + x_2$; et notez que x_1 ne descend pas au-

dessous de 0. Vous pouvez enlever les contraintes de non-négativité en ajoutant la ligne free x_1, x_2 ;

Essayez avec l'objectif modifié.

On obtient une erreur *This problem is unbounded*

Exercice 3

Vérifiez que vous trouvez bien les mêmes solutions à l'exercice 1 de la feuille de TD 1 avec le solveur que celles que l'on a déterminées graphiquement.

On retrouve bien les mêmes solutions que sur le TD1 exercice 1

C'est-à-dire

- $X = 15$ et $y = 10$ pour la première partie
- $X = 9$ et $y = 16$ pour la deuxième partie

2. Un problème générique

Exercice 4

Ecrivez un programme generic.py qui prend en entrée un tel fichier et écrit sur un autre fichier de sortie le programme linéaire associé. Les deux noms de fichier seront fournis sur la ligne de commande

Voir le fichier generic.py

Pour lancer le fichier allez dans le répertoire où se trouve generic.py (Un problème générique/generic.py)

Lancez la commande :

- `python3 generic.py [Nom du fichier entré] [Nom du fichier de sortie]`
- Pour lancer avec l'intégralité
- `python3 generic.py -i [Nom du fichier entré] [Nom du fichier de sortie]`
- `python3 generic.py --int [Nom du fichier entré] [Nom du fichier de sortie]`

- ❖ Les fichiers qu'on a pour mettre en entrée se trouvent dans le sous répertoire (Un problème/générique/data/...

- ❖ Le fichier de sortie sera produit dans le répertoire où se trouve generic.py

Exercice 5

Modifiez le programme generic.py pour qu'il lance également le solveur lp_solve avec le programme linéaire comme entrée, récupère la solution et affiche l'optimum ainsi que les variables non-null et leurs valeurs.

Pour cela on a ajouté la commande : `os.system("lp_solve " + sys.argv[2])` (`argv[2]` étant le fichier de sortie produit)

Exercice 6

Essayez votre programme avec le fichier data.txt.

- Quel est le bénéfice optimal ?
Le bénéfice optimal est : 21654.79332331
- Combien de produits différents faut-il fabriquer ?
D'après le solveur il faut fabriquer 6 produits différents (P10, P13, P22, P23, P27, P29)
- Combien de temps a mis le solveur pour calculer la solution optimale ?
Après avoir utilisé la commande `time` sur le terminal plusieurs fois, on déduit que le solveur a pris entre 0.061 - 0.075 secondes

Exercice 7

Rajoutez les contraintes d'intégralité. Répondez aux questions de l'exercice précédent. Expliquez le résultat.

Essayez votre programme avec le fichier data.txt.

- Quel est le bénéfice optimal ?
Le bénéfice optimal est : 21638.00000000
- Combien de produits différents faut-il fabriquer ?
D'après le solveur il faut fabriquer 12 produits différents (P1, P6, P10, P13, P16, P18, P19, P20, P22, P25, P27, P29)
- Combien de temps a mis le solveur pour calculer la solution optimale ?
Après avoir utilisé la commande `time` sur le terminal plusieurs fois, on déduit que le solveur a pris environ 26.292 secondes

Exercice 8

Essayez maintenant avec le fichier (beaucoup) plus volumineux bigdata.txt, sans et avec les contraintes d'intégralité. Expliquez le résultat.

Sans intégralité

- a. Quel est le bénéfice optimal ?

Le bénéfice optimal est : 5050533.10609335

- b. Combien de produits différents faut-il fabriquer ?

D'après le solveur il faut fabriquer 100 produits différents

- c. Combien de temps a mis le solveur pour calculer la solution optimale ?

Après avoir utilisé la commande time sur le terminal plusieurs fois, on déduit que le solveur à pris environ 1.349 secondes

Avec intégralité

Après plusieurs minutes le résultat ne s'affichait toujours pas on a donc décidé de ne pas continuer, comme la contrainte d'intégralité est bien trop couteuse pour de nombreuses contraintes

3. Un problème de découpe

Exercice 9

Reprenez l'exercice 4 de la feuille de TD 2 sur la découpe de barres d'aluminium et trouvez la solution optimale avec le solveur. Comment faut-il découper les barres de 3m pour minimiser le nombre utilisé ?

Voir Un problème de découpe/ex5.lp

On obtient $x_1 = 33$, $x_2 = 1$, $x_5 = 100$, $x_7 = 43$

Pour :

| taille /type découpage | 0.5 | 1 | 1.2 |
|------------------------|-----|---|-----|
| X1 | 6 | 0 | 0 |
| X2 | 4 | 1 | 0 |
| X3 | 3 | 0 | 1 |
| X4 | 2 | 2 | 0 |
| X5 | 1 | 0 | 2 |
| X6 | 1 | 1 | 1 |
| X7 | 0 | 3 | 0 |

Exercice 10

Ecrivez une fonction qui prend en argument la longueur des barres de base (en cm) ainsi qu'une liste de longueurs souhaitées. La fonction génère tous les différents découpages maximaux possibles.

Exercice 11

En utilisant la fonction de l'exercice précédent, écrivez un programme qui prend en entrées la longueur des barres de base, une liste des longueurs souhaitées et une liste avec le nombre d'éléments de chaque longueur souhaitée. Le programme génère le programme linéaire correspondant et lance le solveur pour trouver l'optimum et la façon dont il faut découper les barres.