Les calculatrices, ordinateurs et documents de cours sont interdits.

I Production de parfum

Une entreprise veut produire des parfums en gagnant le plus possible d'argent :

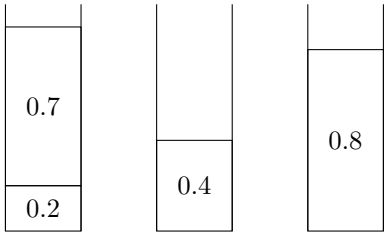
	Produit A (en litres)	Produit B (en litres)	Produit C (en litres)	Prix (€/litre)
Parfum 1	1	0	3	30
Parfum 2	0	2	2	50
Stock	4	12	18	

Par exemple, le parfum 1 demande 1 litre de produit A, 1 litre de produit B et 3 litres de produit C.
De plus, il y a un stock limité de chaque produit (4 litres de produit A, par exemple).

1. Modéliser ce problème sous forme d'un programme linéaire P.
2. Résoudre P graphiquement (donner l'optimum ainsi que la production de chaque parfum pour l'obtenir).
3. Mettre P sous forme canonique P'.
4. Résoudre P' avec l'algorithme du simplexe et vérifier que l'optimum obtenu est le même qu'en question 2.
5. Écrire le dual de P'.

II Problème du *bin packing*

Dans le problème du *bin packing*, on considère n objets dont les poids sont w_1, ..., w_n (qu'on suppose entre 0 et 1), que l'on peut mettre dans des boîtes (*bin*) de capacité 1 (ce qui signifie que la somme des poids mis dans une boîte ne peut pas être supérieure à 1). L'objectif est d'utiliser le moins de boîtes possibles pour ranger tous les objets.
Par exemple, si on a 3 objets de poids 0.2, 0.7, 0.4 et 0.8 alors on peut les ranger avec 3 boîtes : 0.2 et 0.7 dans la première, 0.4 dans la deuxième et 0.8 dans la troisième.



1. Est-ce que 3 boîtes est le minimum possible pour cet exemple ? Justifier.

Voici 3 algorithmes gloutons pour le *bin packing*, qui considèrent les objets un par un :

- *Next-Fit* : Lorsque le prochain objet ne tient pas dans la boîte actuelle, fermer cette boîte et mettre l'objet dans une nouvelle boîte.
- *Next-Fit decreasing* : Même chose que *Next-Fit*, mais en considérant les objets par ordre de poids décroissant.
- *First-fit* : Ajouter le prochain objet à la première boîte (c'est-à-dire la boîte la plus ancienne) dans laquelle il rentre. Si l'objet ne rentre dans aucune boîte, créer une nouvelle boîte et mettre l'objet dedans.
- *Best-fit* : Ajouter le prochain objet à la première boîte avec le plus d'espace libre. Si le objet ne rentre dans aucune boîte, créer une nouvelle boîte et mettre l'objet dedans.

Ainsi, *Next-Fit* considère seulement la dernière boîte créée à chaque itération alors que *First-fit* et *Best-fit* les considèrent toutes.

2. Pour chacun de ces algorithmes gloutons, donner un exemple qui montre que l'algorithme ne donne pas forcément l'optimum. On détaillera l'exécution de l'algorithme à chaque fois.
3. Montrer que le nombre de boîtes donné par *Next-Fit* est au plus deux fois le nombre optimal de boîtes.

On s'intéresse maintenant à un programme linéaire pour le *bin packing*.

4. Écrire un programme linéaire dont l'optimum permet de résoudre le problème du *bin packing*. On pourra remarquer qu'au plus n boîtes sont nécessaires s'il y a n objets, et utiliser des variables $c_i \in \{0,1\}$ déterminant si la boîte i est utilisée.

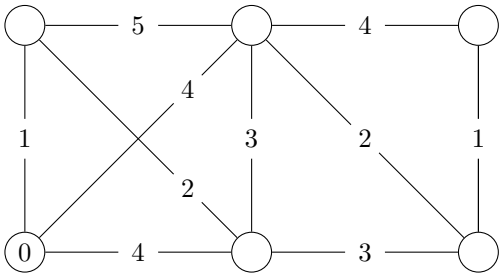
5. Que vaut l'optimum du programme linéaire pour 5 objets tous de poids 0.6 ? Et pour le programme linéaire relaxé ? On donnera les variables du programme linéaire correspondant à cet optimum.

III Problème du voyageur de commerce

Soit $G = (V, E)$ un graphe complet pondéré par $w : E \longmapsto \mathbb{R}^+$.
Un cycle est une suite d'arêtes qui se suivent.
Le fait que G soit complet signifie que toutes les arêtes possibles existent, c'est-à-dire : $E = \{\{u, v\} \mid u, v \in V\}$.

Le problème du voyageur de commerce (TSP : Traveling Salesman Problem) consiste à trouver un cycle de poids minimum visitant tous les sommets exactement une fois.

1. Donner un cycle optimal du TSP sur le graphe G_{ex} ci-dessous (où toutes les arêtes non dessinées sont de poids 100) :



On définit une variable binaire $x_{\{u,v\}}$ pour chaque $\{u, v\} \in E$. Soit P le programme linéaire suivant :

$$\min \sum_{e \in E} w(e) x_e$$

$$\forall u \in V : \sum_{v \in V \setminus \{u\}} x_{\{u,v\}} = 1$$

$$\forall e \in E : \ x_e \in \{0, 1\}$$

2. Expliquer en français quel est le rôle des variables, de la fonction objective et des contraintes de P.

3. Montrer, en donnant un exemple sur G_{ex}, qu'une solution admissible de P (vérifiant les contraintes) ne donne pas forcément un cycle.

On ajoute alors des variables y_v, $\forall v \in V$ et les contraintes suivantes à P :

$$\forall v \in V \setminus \{0\}, \ u \in V \setminus \{0, v\} : \ y_v \geq y_u + (n+1) x_{\{u,v\}} - n \qquad (*)$$

$$\forall v \in V : \ y_v \geq 0$$

On appelle P' ce nouveau programme linéaire.

4. Que devient la contrainte $(*)$ quand $x_{\{u,v\}} = 0$? Quand $x_{\{u,v\}} = 1$?

5. Montrer que le cycle optimal de la question 1 est une solution admissible pour P', pour des valeurs des variables x_e et y_v qu'on donnera.

6. Montrer qu'une solution optimale de P' permet de résoudre le TSP (c'est-à-dire que résoudre P' permet d'obtenir un cycle de poids minimum).