Feuille 1
Modélisation

Exercice 1. *Modélisation d'un problème nutritionnel*

Jacques se demande combien d'argent il doit dépenser en nourriture pour subvenir à ses besoins quotidiens en énergie (2000 kcal), en protéine (55g) et en calcium (800mg). Il choisit 6 aliments qui semblent être des sources bon marché d'éléments nutritifs. Les données nutritionnelles de ces aliments sont reportées dans le tableau ci-dessous.

aliments	taille des portions	énergie (kcal)	protéine (g)	calcium (mg)	prix par portion (centimes d'€)
flocons d'avoine	28 g	110	4	2	11
poulet	100 g	205	32	12	140
oeuf	2 gros	160	13	54	100
lait entier	237 ml	160	8	285	52
tarte aux cerises	170 g	420	4	22	132
porc aux haricots	260 g	260	14	80	105

Sur les conseils d'un nutritioniste, Jacques décide de limiter le nombre de portions de chacun des 6 aliments pour éviter par exemple de prendre 10 portions de porc aux haricots qui couvriraient tous ses besoins pour un prix de 10,5 €. Les limitations sont les suivantes :

flocons d'avoine	:	4 portions au plus par jour
poulet	:	3 portions au plus par jour
oeuf	:	2 portions au plus par jour
lait entier	:	8 portions au plus par jour
tarte aux cerises	:	2 portions au plus par jour
porc aux haricots	:	2 portions au plus par jour

Jacques veut connaître le nombre de portions (les quantités) de chacun des 6 aliments nécessaires pour couvrir tous ses besoins en énergie, en protéines et en calcium, en dépensant le moins d'argent possible (en centimes d'€) et tout en respectant les limitations sur les quantités mentionnées ci-dessus.

1. Modéliser ce problème sous la forme d'un programme linéaire. Ecrire les contraintes sous la forme matricielle $A\mathbf{x} \geq \mathbf{b},\ \mathbf{0} \leq \mathbf{x} \leq \mathbf{d}$ en précisant la matrice A et les vecteurs $\mathbf{b}$ et $\mathbf{d}$.
2. Ecrire le problème de programmation linéaire sous la forme standard $\min_{\tilde{\mathbf{x}}} \mathbf{c}^\top \tilde{\mathbf{x}}$ avec $\tilde{A}\tilde{\mathbf{x}} = \mathbf{b}$, $\tilde{\mathbf{x}} \geq \mathbf{0}$.

Exercice 2. *Le problème du restaurateur*

Un restaurateur a constaté que sa clientèle aime les coquillages. Lorsqu'il en propose dans son restaurant, ses clients les consomment jusqu'à épuisement de sa livraison du jour. Sa carte comporte deux plateaux :

— un plateau "riche" à P_r euros qui comporte n_1^r oursins, n_2^r palourdes, n_3^r huîtres
— un plateau "simple" à P_s euros qui comporte n_1^s oursins, n_2^s palourdes, n_3^s huîtres

1. Un jour donné, son fournisseur lui a livré k_1 oursins, k_2 palourdes et k_3 huîtres.
   Comment doit-il répartir les coquillages entre les deux sortes de plateaux pour réaliser un chiffre d'affaire maximal ? Modéliser ce problème sous forme de programmation linéaire. Ecrire la forme canonique pure et la forme standard du PL.

2. En début de soirée, une de ses amies vient le trouver. Elle a plus d'invités que prévu et manque de coquillages alors que les magasins sont fermés. Elle lui propose de lui racheter tous ses coquillages.

   Quel prix minimum peut-elle proposer pour chaque oursin, chaque palourde et chaque huître afin qu'il ne soit pas tenté d'en conserver une partie pour composer quelques plateaux ? Modéliser ce nouveau problème et comparer-le au précédent.

Exercice 3. *Gestion de projets*

Les n étudiants d'une formation doivent réaliser des projets individuels. Chaque étudiant doit réalisé un et un seul projet. De même, chaque projet doit être attribué à un étudiant et un seul. Pour décider de la répartition des projets, les étudiants choisissent 3 projets et les classent par ordre strictement croissant de préférence. Les projets retenus par un étudiant se voient ainsi attribués une note de 1 à 3 et on considère que les projets non-retenus ont la note 0. On cherche à maximiser la satisfaction générale définie comme étant la somme (ou la moyenne) des notes données par les étudiants aux projets auxquels ils sont affectés.

1. Modéliser ce problème sous forme de programmation linéaire, afin de déterminer une répartition optimale des projets en maximisant la satisfaction générale. Vous introduirez une matrice E des notes projets/étudiants dont les coefficients sont les notes données à chaque projet par chaque étudiant.
2. Pour $n = 4$, donner un exemple de matrice E des notes projets/étudiants. Pour $n = 4$, écrire le problème de programmation linéaire obtenu précédemment, sous la forme

$$\max_{\mathbf{x}} \left[F(\mathbf{x}) = \mathbf{e}^\top \mathbf{x} \right] \quad \begin{cases} A\mathbf{x} = \mathbf{b} \\ \mathbf{x} \in \{0,1\}^{n^2} \end{cases} \tag{1}$$

   en précisant la variable $\mathbf{x}$ et en explicitant les vecteurs $\mathbf{e}$ et $\mathbf{b}$ ainsi que la matrice A.
3. La modélisation précédente souffre d'un manque de considération individuelle... Dans la répartition optimale précédente, des étudiants peuvent très bien se voir attribuer des projets qu'il n'avaient pas classés initialement (i.e. avec la note 0). On décide alors que chaque étudiant doit être affecté à un projet parmi les 3 projets classés initialement.
   (a) Modéliser cette nouvelle contrainte.
   (b) Pour $n = 4$, écrivez cette contrainte sous la forme

$$B\mathbf{x} \geq \mathbf{d} \tag{2}$$

   en explicitant la matrice B et le vecteur $\mathbf{d}$.
   (c) Donner un exemple de matrice des notes projet/étudiant pour laquelle il n'y a pas de solution réalisable au problème (1),(2).

Exercice 4. *Problème d'une entreprise de vols "charter"*

Une compagnie aérienne de vols "charter" doit effectuer sur une plage de temps donné, un certain nombre de vols pour lesquels on connaît :

— AD_i l'aérogare de départ
— AA_i l'aérogare d'arrivée
— HD_i l'heure de départ
— HA_i l'heure d'arrivée

Un avion peut effectuer un vol j à la suite d'un vol i à condition qu'un intervalle de temps suffisant existe entre l'arrivée de i à l'instant HA_i en AA_i et le départ de j à l'instant HD_j en AD_j de manière à permettre éventuellement son transfert en vol à vide de AA_i à AD_j qui dure T_{ij} et la préparation du vol suivant qui dure p. Pour leur premier vol, on suppose les avions disponibles au bon endroit et après leur dernier vol, on les laisse là où ils viennent d'atterrir.

1. Un premier problème à résoudre consiste à minimiser le nombre total d'avions nécessaires. (*Indication.* Minimiser le nombre total d'avions est équivalent à maximiser le nombre de réemplois immédiats car nb de vols = nb d'avions + nb de réemplois et le nombre de vols est fixé.)
2. Un deuxième problème consiste à fixer le nombre d'avions et à minimiser la durée des trajets à vide.

Modéliser ces deux problèmes de programmation linéaire sous les formes canoniques et standards.

Exercice 5. *Stockage*

Une entreprise de service vidéo à la demande (vod) stocke tous ses fichiers vidéos sur un ensemble de disques durs identiques qui ont tous la même taille. L'entreprise souhaite utiliser le minimum de disques durs pour le stockage. Elle dispose de n fichiers vidéo de taille $c_1, \cdots, c_n$ avec m disques durs identiques de taille C. On stocke tous les fichiers sur les disques durs. Pour qu'un stockage soit admissible il faut que la somme des tailles des fichiers stockés sur un disque soit inférieure ou égale à C (on ne peut pas dépasser la capacité d'un disque). De plus, un fichier n'est stocké qu'une seule fois sur un seul disque. Déterminer un stockage consiste alors à déterminer sur quel disque j est stocké le fichier i donné. On cherche à déterminer le stockage (admissible) des fichiers sur les disques durs de façon à minimiser le nombre de disques durs utilisés.

En considérant (entre autres) les variables binaires y_j pour indiquer si le disque dur j est utilisé ou non, modéliser ce problème par un programme linéaire portant uniquement sur des variables binaires.

Exercice 6. *Modélisation du problème du voyageur de commerce par programmation linéaire (Miller-Tucker-Zemlin, 1960).*

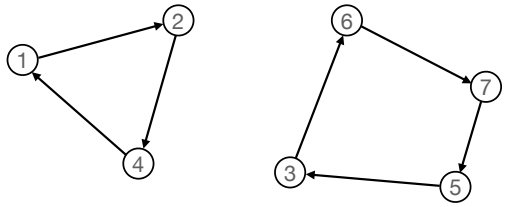
Le problème du voyageur de commerce (TSP Travelling Salesman Problem) s'énonce de la façon suivante. Etant données n villes, trouver le plus court *trajet* passant par toutes les villes une et une seule fois, en revenant à la ville de départ à la fin du trajet. Le but de cet exercice est de modéliser ce problème sous forme d'un programme linéaire. On note $d_{ij} \geq 0$ la distance séparant la ville i de la ville j (a priori $d_{ij} \neq d_{ji}$ pour $i \neq j$ et les valeurs d_{ii} soit valent $+\infty$, soit elles ne seront pas prises en compte dans tout ce qui suit). On introduit les variables d'affectation x_{ij} $(1 \leq i, j \leq n)$ telles que

$$x_{ij} = \begin{cases} 1 & \text{si on va de la ville } i \text{ à la ville } j \\ 0 & \text{sinon.} \end{cases}$$

On suppose que le ville de départ (et d'arrivée) est la ville n°1.

Préliminaires.

1. A partir des variables x_{ij}, définir l'objectif et les contraintes auxquelles sont soumises ces variables.
2. On peut représenter le TSP à l'aide d'un graphe où les sommets correspondent aux villes et les arêtes aux déplacements entre les villes dans le trajet. Dans le TSP, un trajet ne peut être constitué de *sous-circuits* (boucles) non reliés entre eux. Par exemple, le graphe ci-dessous présente 2 sous-circuits.



Pour cet exemple, écrire la matrice $X = (x_{ij})_{1 \leq i,j \leq 7}$ et montrer que les contraintes sur ces variables sont respectées. Cette solution réalisable ne respecte pourtant pas la définition d'un trajet.

Modélisation MTZ. Pour éviter de possibles *sous-circuits*, on introduit les variables supplémentaires $u_i \in \mathbb{N}$ pour $1 \leq i \leq n$, représentant la position de la ville i dans la tournée du voyageur. Comme on a supposé que la ville 1 est la ville de départ, on a donc $u_1 = 1$ et par ailleurs $u_i \in \{2, \cdots, n\}$ pour $i \in \{2, \cdots, n\}$.

3. Les variables x_{ij} et u_i sont liées par le fait que si $x_{ij} = 1$ alors nécessairement $u_i < u_j$. Montrer que cette contrainte peut se traduire algébriquement par

$$u_i - u_j + 1 \leq \alpha(1 - x_{ij}), \quad 2 \leq i \neq j \leq n \tag{1}$$

   où α est un réel à déterminer.

4. Modéliser le problème du voyageur de commerce sous forme d'un programme linéaire pour les variables x_{ij} $(1 \leq i \neq j \leq n)$ et u_i $(2 \leq i \leq n)$.

5. Dans l'exemple précédent présentant deux sous-circuits, montrer que les contraintes (1) ne peuvent pas être toutes vérifiées en faisant la somme de ces contraintes pour le sous-circuit $(3, 6, 7, 5, 3)$.

6. De façon générale, on peut montrer que s'il existe un sous-circuit alors nécessairement celui-ci doit passer par la ville de départ nº1. Ceci implique qu'il ne peut exister qu'un seul circuit partant de la ville 1, passant par toutes les villes et revenant à la ville 1 de départ (un *trajet*). Pour établir cette propriété, supposer par l'absurde qu'il existe un sous-circuit avec k villes $(i_1, i_2, \cdots, i_k, i_1)$ qui ne passe pas par la ville 1 de départ (i.e $i_j \neq 1$ pour $j = 1, \cdots, k$). Montrer alors une contradiction avec les contraintes (1) concernées.

Exercice 7. *Sudoku*

Un sudoku est composé d'une grille de 3×3 sous-grilles elles-mêmes constituées de 3×3 cases, soit au total 81 cases. Le but du jeu est de remplir la grille avec une série de chiffres allant de 1 à 9 tous différents, qui ne se trouvent jamais plus d'une fois sur une même ligne, dans une même colonne ou dans une même sous-grille. Quelques chiffres sont déjà placés dans la grille.

2	·	·	·	·	**4**	·	**3**	·
·	**9**	·	·	·	·	·	**7**	·
·	·	·	·	**6**	·	·	**4**	·
3	**8**	·	·	·	**7**	·	·	·
6	·	·	·	·	·	**5**	·	·
·	·	**2**	·	·	·	**8**	·	·
·	·	·	**8**	·	·	**9**	·	·
·	·	·	·	**9**	·	·	·	**2**
·	·	**4**	·	·	**2**	·	**8**	·

FIGURE 1 – Exemple d'une grille de sudoku

L'objectif de cet exercice est de modéliser le jeu de Sudoku par programmation linéaire en nombres entiers. Dans ce but, on introduit les variables binaires d'affectation suivantes : pour une case $(i, j) \in [\![1, 9]\!]^2$,

$$x_{ijk} = \begin{cases} 1 & \text{si le chiffre } k \in [\![1, 9]\!] \text{ est affecté à la case } (i, j) \\ 0 & \text{sinon} \end{cases} \tag{2}$$

1. Ecrire les différentes contraintes de la règle du jeu comme des contraintes algébriques linéaires portant sur les variables x_{ijk}.

2. Dans le problème de Sudoku, il n'y a rien à optimiser, il n'y a que des contraintes. On peut cependant fixer l'objectif $f \equiv 1$ ou bien introduire des variables artificielles a_l dans les contraintes et l'objectif devient $\min \sum_l a_l$. Ecrire le problème de Sudoku comme un PL.

Exercice 8. *Affectation de bureaux*

Une entreprise va emménager dans de nouveaux locaux. Ceux-ci se composent de n bureaux identiques. Chacun de ces bureaux peut accueillir indistinctement l'un des n services de l'entreprise. Chaque bureau accueille un et un seul service. De plus, chaque service est accueilli entièrement dans un et un seul bureau. La disposition des bureaux est connue et on note d_{jl} la distance entre les bureaux j et l. L'entreprise a effectué dans les anciens locaux des statistiques sur les va-et-vient entre les n services à installer dans les n bureaux. Elle dispose du nombre c_{ik} de fois où des employés vont du service i au service k et où des employés vont du service k au service i. L'entreprise souhaite affecter les services aux bureaux de sorte que la distance totale parcourue par les employés soit la plus petite possible.

1. Combien y-a-t-il de possibilités d'affecter les n services aux n bureaux ? L'ordinateur dont vous disposez peut évaluer 1 milliard de possibilités à la seconde. Pour $n = 30$, estimer le temps mis pour obtenir par une méthode exhaustive (c'est-à-dire en essayant toutes les solutions), l'une des meilleures configurations donnant la distance totale parcourue minimale (*indication :* $30! \simeq 10^{32}$).
2. Modéliser le problème qui consiste à placer les services dans les bureaux de manière à minimiser la somme des distances parcourues par les employés. Vous pourrez introduire des variables binaires x_{ij} doublement indicées. Vous obtiendrez ainsi un problème de programmation *quadratique* sur ces variables.
3. *Linéarisation.* Pour linéariser le problème quadratique précédent, on introduit de nouvelles variables binaires à 4 indices $y_{ijkl} = x_{ij}x_{kl}$. Montrer que

$$y_{ijkl} = x_{ij}x_{kl} \quad \Leftrightarrow \quad \begin{cases} y_{ijkl} \leq x_{ij} \\ y_{ijkl} \leq x_{kl} \\ x_{ij} + x_{kl} \leq 1 + y_{ijkl} \end{cases}$$

Ecrire le problème de programmation *linéaire* correspondant.

Exercice 9. *Identification de paramètres et programmation linéaire*

On considère deux grandeurs physiques t et y qui sont liées entre-elles par une relation fonctionnelle du type $y = f(t)$. On dispose d'un modèle pour la fonction f. On sait que f est une combinaison linéaire de n fonctions élémentaires $\phi_1, \cdots, \phi_n$ connues :

$$f(t) = f_{\mathbf{x}}(t) = x_1\phi_1(t) + \cdots + x_n\phi_n(t)$$

où $\mathbf{x} = (x_1, \cdots, x_n)^\top \in \mathbb{R}^n$ sont des paramètres réels qu'on cherche à identifier. Pour cela, on réalise une série de m mesures $(t_i, y_i)_{1 \leq i \leq m}$ et on cherche à minimiser le plus grand écart absolu entre le modèle donné par $f_{\mathbf{x}}$ et les mesures :

$$\min_{\mathbf{x} \in \mathbb{R}^n} \max_{1 \leq i \leq m} |f_{\mathbf{x}}(t_i) - y_i| \tag{3}$$

Remarques : si l'on remplace l'objectif (3) par la somme des carrés des écarts (i.e. $\min_{\mathbf{x} \in \mathbb{R}^n} \sum_{i=1}^{m} |f_{\mathbf{x}}(t_i) - y_i|^2$, alors on obtient exactement la formulation en moindres carrés...).

On notera $\|\mathbf{x}\|_\infty = \max_{1 \leq i \leq n} |x_i|$ et $\|\mathbf{x}\|_1 = \sum_{i=1}^{n} |x_i|$ les normes sur $\mathbb{R}^n$ d'un vecteur $\mathbf{x} = (x_1, \cdots, x_n)^\top$.

1. Ecrire le problème (3) sous la forme

$$\min_{\mathbf{x}\in\mathbb{R}^n} \|A\mathbf{x}-\mathbf{b}\|_\infty := \min_{\mathbf{x}\in\mathbb{R}^n} \max_{1\le i\le m} |A\mathbf{x}_i - b_i| \tag{4}$$

   en précisant ce que valent le vecteur $\mathbf{b} = (b_1, \cdots, b_m)^\top$ et la matrice $A \in \mathcal{M}_{m\times n}$.

2. Transformer le problème précédent en un problème de programmation linéaire sous forme canonique pure. (*Indication :* introduire la variable $e = \max_i |A\mathbf{x}_i - b_i|$).

3. On considère à présent la fonction objectif

$$\min_{\mathbf{x}\in\mathbb{R}^n} \|A\mathbf{x}-\mathbf{b}\|_1 := \min_{\mathbf{x}\in\mathbb{R}^n} \sum_{i=1}^{m} |A\mathbf{x}_i - b_i| \tag{5}$$

   Transformer ce nouveau problème en un problème de programmation linéaire sous forme canonique pure.