

FEUILLE 1 : MODÉLISATION

**Exercice 1** *Résolution graphique d'un programme linéaire (pb de production)*

Une usine fabrique deux produits  $P_1$  et  $P_2$  en utilisant 4 machines  $M_1, \dots, M_4$ . Chacun de ces deux produits nécessite un temps de passage donné (en heures) sur chacune des 4 machines. Les temps de passage par unité de produit sont indiqués dans le tableau ci-dessous. Par ailleurs, chaque machine n'est disponible que pendant une certaine durée, également indiquée dans le tableau.

	$M_1$	$M_2$	$M_3$	$M_4$
$P_1$	0h	1,5h	2h	3h
$P_2$	3h	4h	3h	0h
disponibilité	39h	60h	57h	57h

Les produits  $P_1$  et  $P_2$  rapportent respectivement les bénéfices  $B_1 = 1700$  euros et  $B_2 = 3200$  euros par unité. On cherche à déterminer le plan de production pour maximiser le bénéfice total.

1. Modéliser ce problème sous forme de programmation linéaire (PL). Donner la forme canonique pure et la forme standard du programme linéaire.
2. Résoudre **graphiquement** le PL.

**Exercice 2** *Résultat utile pour certains exercices*

On considère le problème d'optimisation :

$$(\mathcal{P}) : \min_{x \in E} f(x), \text{ avec } f(x) = \max_{i \in \llbracket 1, m \rrbracket} f_i(x)$$

où  $E \subset \mathbb{R}^n$  et pour chaque  $i$ ,  $f_i : E \rightarrow \mathbb{R}$ . Rmq : ce problème est déjà contraint si  $E \neq \mathbb{R}^n$ , par exemple  $E$  pourrait être un polyèdre de  $\mathbb{R}^n$ .

On considère maintenant le problème d'optimisation :

$$(\mathcal{Q}) : \begin{cases} \min_{(x,e) \in E \times \mathbb{R}} g(x,e) = e \\ f_i(x) \leq e, \forall i \in \llbracket 1, m \rrbracket \text{ (soit } m \text{ contraintes supplémentaires.)} \end{cases}$$

Montrer que ces deux problèmes sont équivalents au sens suivant : si  $x^*$  est optimale pour  $(\mathcal{P})$  alors  $(x^*, e^*)$  où  $e^* = \max_{i \in \llbracket 1, m \rrbracket} f_i(x^*)$  est optimale pour  $(\mathcal{Q})$  et inversement si  $(x^*, e^*)$  est optimale pour  $(\mathcal{Q})$  alors  $x^*$  est optimale pour  $(\mathcal{P})$ .

**Exercice 3** *Parallélisme et répartition de tâches*

Un programme consiste à exécuter  $n$  procédures (tâches) indépendantes les unes des autres. Pour cela on dispose de  $m$  processeurs et on sait que le temps de calcul de la procédure  $i$  effectuée sur le processeur  $j$  est égal à  $c_{ij} > 0$ <sup>1</sup>. Chacune des  $n$  tâches doit être effectuée par un et un seul processeur. Mais chaque processeur peut exécuter plusieurs tâches ou aucune. On cherche à minimiser le temps de calcul, c'est à dire le temps entre le lancement et celui où la dernière tâche est terminée.

*Modéliser ce problème d'optimisation en faisant intervenir des variables binaires. Pour déterminer la fonction objectif, on pourra s'intéresser d'abord au temps  $T_j$  passé sur le processeur  $j$ . On remarquera que ce problème n'est pas directement un PL mais peut se reformuler comme tel en utilisant le résultat précédent.*

---

1. Rmq : les processeurs ne sont pas forcément identiques et donc  $c_{ij}$  n'est pas forcément égal  $c_{ij'}$  si  $j \neq j'$ .

#### Exercice 4 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  $c_1, \dots, c_n$  qu'elle souhaite stocker sur des disques durs tous de taille identique égale à  $C$ . En utilisant une heuristique, on a trouvé que  $m$  disques durs suffisaient pour ce stockage mais cette valeur n'est peut-être pas optimale (i.e. on peut peut-être utiliser moins de  $m$  disques).

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 5 Gestion de projet (extrait du partiel de GRO 2012)

Les élèves d'une célèbre école d'ingénieurs de l'Est de la France doivent réaliser en 2ème année un Projet d'Introduction Douleuruse à la Recherche (PIDR). L'enseignant responsable des Projets est chargé de répartir les projets entre élèves. Il y a autant d'élèves que de projets et chaque élève doit être affecté à un projet et un seul. De même, chaque projet doit être attribué à un élève et un seul. Il y a  $n$  projets ( $n \geq 3$ ). Pour décider de la répartition des projets, l'enseignant demande à chaque élève de choisir 3 projets parmi les  $n$  et de les classer par ordre strictement croissant de préférence. Chaque projet retenu par un élève se voit ainsi attribué une note de 1 à 3 et on considère que les projets non-retenus par l'élève ont la note 0. L'idée de l'enseignant (qui veut oeuvrer pour le bien collectif) est de maximiser la satisfaction générale. Cette satisfaction est définie comme étant la somme (ou la moyenne) des notes données par les élèves aux projets auxquels ils sont affectés. Il vous est demandé de venir en aide à l'enseignant en mettant en oeuvre toutes vos compétences acquises en programmation linéaire.

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/élèves dont les coefficients sont les notes données à chaque projet par chaque élève.
2. Pour  $n = 4$ , donner un exemple de matrice  $E$  des notes projets/élèves. Pour  $n = 4$ , écrire le problème de programmation linéaire obtenu précédemment, sous la forme

$$\begin{aligned} \max_{\mathbf{x}} & [F(\mathbf{x}) = \mathbf{e}^\top \mathbf{x}] \\ \text{s.t.} & \begin{cases} A\mathbf{x} = \mathbf{b} \\ \mathbf{x} \in \{0, 1\}^{n^2} \end{cases} \end{aligned} \quad (1)$$

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 élèves peuvent très bien se voir attribuer des projets qu'il n'avaient pas classés initialement (i.e. avec la note 0). L'enseignant décide alors que chaque élève 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

$$B\mathbf{x} \geq \mathbf{d} \quad (2)$$

en explicitant la matrice  $B$  et le vecteur  $\mathbf{d}$ .

- (c) Donner un exemple de matrice des notes projet/élève pour laquelle il n'y a pas de solution réalisable au problème (1),(2).

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

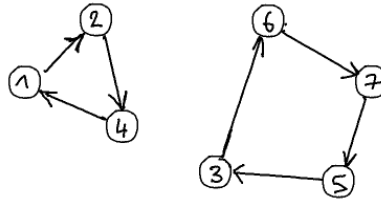
Le problème du voyageur de commerce (TSP Travelling Salesman Problem) s'énonce de la façon suivante. Étant données  $n$  villes, trouver le plus court *trajet* passant par toutes les villes une et une seule fois, en revenant à la ville de départ à la fin du trajet. Le but de cet exercice est de modéliser ce problème sous forme d'un programme linéaire. On note  $d_{i,j} \geq 0$  la distance séparant la ville  $i$  de la ville  $j$  (avec a priori  $d_{i,j} \neq d_{j,i}$ ) pour  $i \neq j$ . On introduit les variables d'affectation  $x_{i,j}$  ( $1 \leq i, j \leq n$  et  $i \neq j$ ) telles que

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

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

#### Préliminaires.

1. A partir des variables  $x_{i,j}$ , 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 le tableau<sup>3</sup>  $X = (x_{i,j})_{1 \leq i, j \leq 7; i \neq j}$  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  $u_i \in \{2, \dots, n\}$  pour  $i \in \{2, \dots, n\}$ .

3. Les variables  $x_{ij}$  et  $u_i$  sont liées par le fait que si  $x_{ij} = 1$  alors nécessairement<sup>4</sup>  $u_i < u_j$ . Montrer que cette contrainte peut se traduire algébriquement par :

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

où  $\alpha$  est un réel à déterminer.

4. Modéliser le problème du voyageur de commerce sous forme d'un programme linéaire pour les variables  $x_{ij}$  ( $1 \leq i, j \leq n; i \neq j$ ) 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 n°1. 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 ( $i_1, i_2, \dots, i_k, i_1$ ) qui ne passe pas par la ville 1 de départ (i.e  $i_j \neq 1$  pour  $j = 1, \dots, k$ ). Montrer alors une contradiction avec les contraintes (1) concernées.

2. En particulier si on doit faire une tournée en ville (les  $n$  villes étant alors  $n$  adresses dans la ville) il est clair que  $d_{i,j} \neq d_{j,i}$  du fait des sens interdits. Si les villes sont assez éloignées on peut effectivement faire l'hypothèse que  $d_{i,j} = d_{j,i}$ .

3. Mettre éventuellement une croix dans les cases diagonales qui n'existent pas.

4. En fait si  $x_{i,j} = 1$  alors  $u_j = u_i + 1$  mais utiliser exactement cette information ne conduit pas à un PL.

**Exercice 8** 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  $\phi_1, \dots, \phi_n$  connues :

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

où  $\mathbf{x} = (x_1, \dots, x_n)^\top \in \mathbb{R}^n$  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 :

$$\min_{\mathbf{x} \in \mathbb{R}^n} \max_{1 \leq i \leq m} |f_{\mathbf{x}}(t_i) - y_i| \quad (2)$$

*Remarques* : si l'on remplace l'objectif (2) 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  $\mathbf{x} = (x_1, \dots, x_n)^\top$ .

1. Écrire le problème (2) sous la forme

$$\min_{\mathbf{x} \in \mathbb{R}^n} \|\mathbf{Ax} - \mathbf{b}\|_\infty := \min_{\mathbf{x} \in \mathbb{R}^n} \max_{1 \leq i \leq m} |\mathbf{Ax}_i - b_i| \quad (3)$$

en précisant ce que valent le vecteur  $\mathbf{b} = (b_1, \dots, b_m)^\top$  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  $e = \max_i |\mathbf{Ax}_i - b_i|$  et utiliser le résultat de l'exercice 2).

**Exercice 9** Sudoku

Un sudoku est composé d'une grille de  $3 \times 3$  sous-grilles elles-mêmes constituées de  $3 \times 3$  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.

<b>2</b>	.	.	.	.	.	<b>4</b>	.	<b>3</b>	.
.	<b>9</b>	.	.	.	.	.	.	<b>7</b>	.
.	.	.	.	.	<b>6</b>	.	.	<b>4</b>	.
<b>3</b>	<b>8</b>	.	.	.	.	<b>7</b>	.	.	.
<b>6</b>	.	.	.	.	.	.	<b>5</b>	.	.
.	.	<b>2</b>	.	.	.	.	<b>8</b>	.	.
.	.	.	<b>8</b>	.	.	.	<b>9</b>	.	.
.	.	.	.	<b>9</b>	.	.	.	.	<b>2</b>
.	.	<b>4</b>	.	.	<b>2</b>	.	<b>8</b>	.	.

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  $(i, j) \in \llbracket 1, 9 \rrbracket^2$ ,

$$x_{ijk} = \begin{cases} 1 & \text{si le chiffre } k \in \llbracket 1, 9 \rrbracket \text{ est affecté à la case } (i, j) \\ 0 & \text{sinon} \end{cases} \quad (4)$$

1. Écrire 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  $\min \sum_l a_l$ . Écrire le problème de Sudoku comme un PL.