M2ISN, RO: Optimisation discrète

Bernhard Beckermann
Labo Painlevé (AN-EDP), ULille
Bernhard.Beckermann@univ-lille.fr

Villeneuve d'Ascq, 11 octobre 2025

Variables astreintes d'être entières

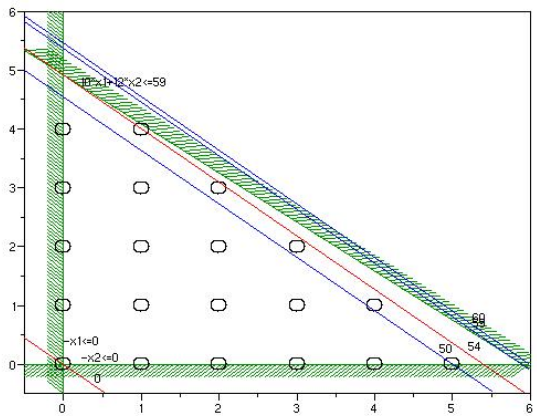
Les problèmes concrets présent souvent des variables qui doivent prendre des valeurs discrètes et pas continues:

- variables de décision dites **bivalentes**: faire ou pas faire, investir ou non
- variables désignant la quantité d'un objet indivisible: nombre de personnes,...

L'exemple suivant montre que SIMPLEX plus arrondi donne des résultats trompeurs: les techniques d'optimisation linéaire continue ne sont plus applicables.

$$\begin{cases} \max(10x_1 + 11x_2) \\ 10x_1 + 12x_2 \leqslant 59, \\ x_1, x_2 \geqslant 0 \text{ entiers.} \end{cases}$$

Solution : $(1, 4)^t$
Solution continue: $(5.9, 0)^t$



Le problème du sac-à-dos

Un sac a une contenance donnée (volume ou poids), et il peut être rempli d'objets d'encombrement fixé, sans jamais dépasser la capacité totale. Le choix pour un objet j est donc prendre ($x_j = 1$ ou pas prendre ($x_j = 0$), connaissant la valeur économique attribué à chaque objet (par exemple survie en montagne). On obtient alors le problème

$$\begin{cases} \max\{f^1 x_1 + ... + f^n x_n\} \\ B^1 x_1 + ... + B^n x_n \leqslant b \\ \forall j : x_j \in \{0, 1\} \end{cases} \qquad \text{c'est-à-dire, } B \in \mathbb{R}^{1 \times n}.$$

Il y a alors 2^n possibilités de choisir la variable x, et pour chaque possibilité il faudra tester la faisabilité et calculer la valeur.
• une simple énumération de toutes les possibilités prend trop de temps.

Variante aux variables doublement bornées: on se permet d'amener plusieurs copies du même objet, la contrainte $x_j \in \{0, 1\}$ devient alors $0 = \underline{x}_j \leqslant x_j \leqslant \overline{x}_j$ et $x_j \in \mathbb{Z}$.

Obtenir des valeurs entières par chance

Pour le problème

$$(P): \quad \max\{fx : x \in \mathbb{R}^n, Ax = a, \underline{x} \leqslant x \leqslant \overline{x}\}$$

si par hasard tous les points de base sont à composantes entières, alors une solution optimale de (P) est aussi une solution optimale du problème obtenu en ajoutant la contrainte $x \in \mathbb{Z}^n$.

Lemme: Si $a, \underline{x}, \overline{x}$ sont à coefficients entiers, et A est totalement unimodulaire, c'est-à-dire, le déterminant de chaque sous-matrice carrée vaut $0, 1$ ou -1, alors tout point de base est à coefficients entiers.

Lemme: Une matrice comportant par colonne au plus deux éléments non nuls, et ces deux éléments étant des éléments distincts de $\{-1, 1\}$, est totalement unimodulaire.

Exemple d'une matrice unimodulaire: problème de transbordement

On se donne trois ensembles d'indices disjoints

L_1 les usines: b_ℓ unités sont disponibles à l'usine $\ell \in L_1$

L_2 les entrepôts : b_ℓ limite de stockage pour l'entrepôt $\ell \in L_2$.

L_3 les clients : b_ℓ unités sont demandés par le client $\ell \in L_3$

Une usine peut livrer aux entrepôts, mais aussi directement à certains clients. Toute quantité arrivant à un entrepôt doit être acheminée vers un client. On dispose alors d'un certain nombre de routes $J = (L_1 \times L_2) \cup (L_1 \times L_3) \cup (L_2 \times L_3)$, et pour chaque route $j \in J$ d'un coût unitaire de transport f^j. On se demande quelle quantité x_j il faut acheminer sur la route j pour satisfaire les clients au moindre coût.

Modélisation

$$\begin{cases} \min \sum_{j\in J} f^j x_j \\ \forall j \in J : \quad x_j \geqslant 0, x_j \in \mathbb{Z} \\ \forall \ell \in L_1 : \quad \sum_{k\in L_2\cup L_3} x_{\ell,k} \leqslant b_\ell \\ \forall k \in L_3 : \quad \sum_{\ell\in L_1\cup L_2} x_{\ell,k} \geqslant b_k \\ \forall k \in L_2 : \quad \sum_{\ell\in L_1} x_{\ell,k} \leqslant b_k \\ \forall k \in L_2 : \quad \sum_{\ell\in L_1} x_{\ell,k} = \sum_{i\in L_3} x_{k,i} \end{cases}$$

Après introduction de variables d'écart, la matrice A est totalement unimodulaire si L_2 est vide: le **problème de transport**.
En introduisant les nouvelles variables doublement bornées

$$\forall k \in L_2 : \quad \gamma_k = \sum_{\ell\in L_1} x_{\ell,k}$$

on peut toujours se ramener à une matrice totalement unimodulaire.

Le problème du voyageur de commerce

N villes énumérés $1, ..., n$, chacune reliée aux autres par une route de longueur $f^{j,k}$, doivent être visitées par un voyageur de commerce une fois et une seule. On cherche à déterminer la tournée (i.e., une permutation sur les indices des villes) minimisant la longueur des trajets. Encore une fois, il est hors de question de tester l'ensemble des tournées, c'est-à-dire, $n!$ possibilités. En termes de graphes, on cherche un circuit passant par toutes les villes (dit circuit hamiltonien), de longueur minimum.

Simulation sous Matlab (algorithme heuristique)

En posant la variable décisionnelle $x_{j,k} = 1$ si on emprunte la route (j, k) (de sens unique), on se ramène a des contraintes de type transport:

$$\forall j = 1, ..., n : \quad \sum_{k=1}^{n} x_{j,k} = 1, \quad \sum_{k=1}^{n} x_{k,j} = 1.$$

Modélisation sous AMPL, voir data/hamilton

Le problème est qu'avec nos contraintes de type transport on n'exclut pas des permutations décomposables: un ensemble de plusieurs cycles disjoints.
On introduira alors pour chaque ville j une variable entière $u_j \in \{1, 2, ..., n\}$ (ici $1 \leqslant u_j \leqslant n, u_j \in \mathbb{Z}$) indiquant quand on va visiter la ville j.
Notons que, sans perte de la généralité, la ville de départ est la ville 1: $u_1 = 1$. Aussi, $u_i - u_j + 1 \leqslant n$.

Solution 1: Si $x_{i,j} = 1$ et $j \neq 1$ alors $u_i - u_j + 1 = 0$.
Solution 2: Si $x_{i,j} = 1$ et $j \neq 1$ alors $u_i - u_j + 1 \leqslant 0$.
Effectivement, supposons par absurde l'existence d'un circuit $(i_1, i_2, ..., i_r, i_{r+1} = i_1)$ $(x_{i_\ell, i_{\ell+1}} = 1)$ ne comportant pas la ville 1, alors en formant la somme des inégalités ci-dessus on obtient $\sum_{\ell=1}^{r}(u_{i_\ell} - u_{i_{\ell+1}}) + r = r \leqslant 0$.

Une petite table de vérité montre que la deuxième solution est réalisée par l'ensemble des inégalités

$$\forall (i,j) \text{ route avec } j \neq 1: \quad u_i - u_j + 1 \leqslant (1 - x_{i,j})n.$$

La méthode Branch and Bound
= exploration par séparation et évaluation

Position du problème

On souhaite résoudre

$$v(P \cap Q) := \min\{f\,x : x \in P \cap Q\}$$

avec (par exemple)

- Polyèdre $P = P(\underline{x}, \overline{x}) = \{x \in \mathbb{R}^n : Ax = a, x \geqslant \underline{x}, x \leqslant \overline{x}\}$ (A, a fixés).
- $Q = \mathbb{Z}^n$ ou $Q = \{0,1\}^n$ ou ... traduisant les contraintes "compliquées"
- $P \cap Q$ fini mais "grand": *optimisation combinatoire*
- pour P polyèdre:
  - on peut trouver sans "trop" d'effort $v(P) = f\,x^P$ avec $x^P \in P$ (Simplex)
  - on dispose "facilement" des sous-estimations $g(P)$ avec $g(P) \leqslant v(P)$ (post optimisation)

Exemple: le sac à dos aux variables bivalentes

$$\min\left\{f^1 x_1 + ... + f^n x_n : C_1^1 x_1 + ... + C_1^n x_n \leqslant c_1, x_j \in \{0,1\}\right\}.$$

Un exemple d'une arborescence complète

Une idée peut consister à fixer quelques variables (cas des variables bivalentes) et d'adapter les autres au mieux. Si on utilise cette idée récursivement, on obtient le schéma suivant pour trois variables bivalentes

$$
\begin{array}{lll}
 & P, x_1 = 0 & P, x_1 = 0, x_2 = 0 \\
 & & P, x_1 = 0, x_2 = 1 \\
P & P, x_1 = 1 & P, x_1 = 1, x_2 = 0 \\
 & & P, x_1 = 1, x_2 = 1
\end{array}
$$

$$
\begin{array}{l}
P, x_1 = 0, x_2 = 0, x_3 = 0 \\
P, x_1 = 0, x_2 = 0, x_3 = 1 \\
P, x_1 = 0, x_2 = 1, x_3 = 0 \\
P, x_1 = 0, x_2 = 1, x_3 = 1 \\
P, x_1 = 1, x_2 = 0, x_3 = 0 \\
P, x_1 = 1, x_2 = 0, x_3 = 1 \\
P, x_1 = 1, x_2 = 1, x_3 = 0 \\
P, x_1 = 1, x_2 = 1, x_3 = 1
\end{array}
$$

Variables astreintes d'être entières? On sépare suivant les inégalités du type $x_j \leqslant \nu$, $x_j \geqslant \nu + 1$.

Construire toute l'arborescence n'est généralement pas possible...

Pari du BB (="branch and bound"):

On explore l'arborescence suivant une stratégie qui devrait permettre d'explorer seulement un sous-ensemble "petit". En chaque itération

- une fonction de choix nous dit quel sous-domaine P' il faut traiter;
- une procédure d'évaluation nous dit quoi faire:
- si le sous-problème n'est pas prometteur on peut définitivement élaguer la branche de l'aborescence;
- certains sous-problèmes nous donnent des candidats pour la solution optimale du problème initial, sans devoir explorer de plus près cette branche. Par conséquent, on peut également élaguer la branche de l'aborescence;
- pour les sous-problèmes restants, une fonction de séparation nous dit comment couper P' en deux: on créera deux nouvelles feuilles dans l'arborescence.

  L'idée est de créer des fils P'_1, P'_2 semblables à leur père P': on résout les problèmes associés aux fils par Simplex dual en partant de la solution du père.

... on espère de pouvoir élaguer un maximum de branches, mais nos stratégies seront en grande partie de nature heuristique...

Invariantes: On dispose d'un candidat $x^c \in Q \cap P$ et de sa valeur $F = f\,x^c$, et d'une pile $J = \{P_1, P_2..., P_\ell\}$ de polyèdres $P_j \subset P$ pas encore évalués;
Initialisation: $J \leftarrow \{P\}$, $F \leftarrow +\infty$
Itération:

- On rétire de la pile $P' \in J$ (fonction de *choix*, peut utiliser g);
- On évalue $v(P') = f\,x^{P'}$
- si $v(P') \geqslant F$ (inclus $P' = \emptyset$): ici $\forall\, x \in P' \cap Q : f\,x \geqslant v(P') \geqslant F$. Branche pas prometteuse, on peut l'élaguer.
- si $v(P') < F$ et $x^{P'} \in Q$ (par *chance* ou parce que P' est "petit"): ici $v(P') = v(Q \cap P')$.
  $x^c \leftarrow x^{P'}$, $F \leftarrow v(P')$. Branche complètement explorée, on peut l'élaguer.
- si $v(P') < F$ et $x^{P'} \in P' \setminus Q$: création de deux fils P'_1, P'_2 dans l'arborescence. La fonction de *séparation* nous donne un indice t (en fonction de $x^{P'}$ et l'évaluation rapide g) et on définit

$$P'_1 = \{x \in P' : x_t \leqslant \lfloor x_t^{P'} \rfloor\}, \quad P'_2 = \{x \in P' : x_t \geqslant \lfloor x_t^{P'} \rfloor + 1\}.$$

N.B.1: On peut aussi d'abord calculer $g(P')$, et calculer $v(P')$ et $x^{P'}$ seulement si $g(P') < F$.

N.B.2: Pour les problèmes bivalentes, on fixe la composante x_t pour les fils.

Un exemple numérique

$\min(-2x_1 - x_2)$, $x_1 - 4x_2 \leqslant 0$, $3x_1 + 4x_2 \leqslant 15$, $x_1, x_2 \geqslant 0$ (polyèdre P), $x_1, x_2 \in \mathbb{Z}$ (contraintes $Q = \mathbb{Z}^2$).

$P_1 = P = P(\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}+\infty\\+\infty\end{bmatrix})$, $x^{P_1} = \begin{bmatrix}\frac{15}{4}\\\frac{15}{16}\end{bmatrix}$,

$v(P_1) = -\frac{135}{16} \approx -8.4 \Longrightarrow t = 1$, création de P_2, P_3

$P_2 = \{x \in P_1 : x_1 \leqslant 3\} = P(\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}3\\+\infty\end{bmatrix})$, $x^{P_2} = \begin{bmatrix}3\\\frac{3}{2}\end{bmatrix}$,

$v(P_2) = -\frac{15}{2} = -7.5 \Longrightarrow t = 2$, création de P_4, P_5

$P_4 = \{x \in P_2 : x_2 \leqslant 1\} = P(\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}3\\1\end{bmatrix})$, $x^{P_4} = \begin{bmatrix}3\\1\end{bmatrix}$,

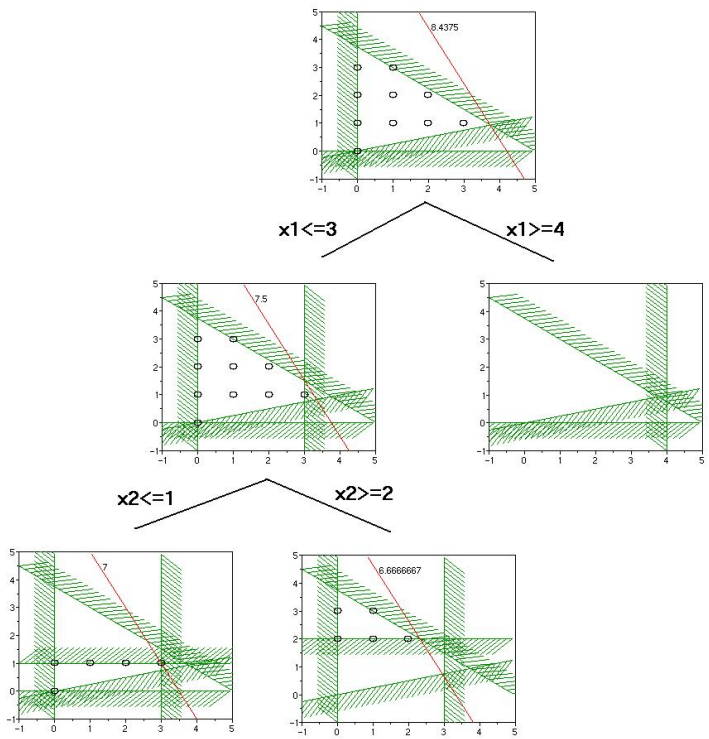
$v(P_4) = -7 \Longrightarrow x^c = x^{P_4}, F = -7$, on élague

$P_5 = \{x \in P_2 : x_2 \geqslant 2\} = P(\begin{bmatrix}0\\2\end{bmatrix}, \begin{bmatrix}3\\+\infty\end{bmatrix})$, $x^{P_5} = \begin{bmatrix}\frac{7}{3}\\2\end{bmatrix}$,

$v(P_4) = -\frac{20}{3} \approx -6.6 > F \Longrightarrow$ branche non prometteuse,

$P_3 = \{x \in P_2 : x_2 \geqslant 2\} = P(\begin{bmatrix}4\\0\end{bmatrix}, \begin{bmatrix}+\infty\\+\infty\end{bmatrix})$,

branche avec polyèdre vide, on élague



Solution optimale dernière solution candidate: $x^c = x^{P_4} = \begin{bmatrix}3\\1\end{bmatrix}$.

Les fonctions de sélection

On retire le candidat P' au début de la file d'attente, qui est organisée comme suit:

"FIFO" (=first in first out): on ajoute les nouveaux polyèdres à la fin de la liste = parcourir l'arborescence en largeur (parallelisable?).

"FILO" (=first in last out): on ajoute les nouveaux polyèdres au début de la liste = parcourir l'arborescence en profondeur (avantage gestion mémoire?)

"best": D'abord FILO pour obtenir "vite" $F < +\infty$, et ensuite un classement croissant des candidats en fonction de leur évaluation rapide g (idée: obtenir vite une valeur F "petite" pour pouvoir élaguer plus de branches?)

On calcule l'évaluation rapide pour les fils

$$P'_1 = \{x \in P' : x_t \leqslant \lfloor x_t^{P'} \rfloor\}, \quad P'_2 = \{x \in P' : x_t \geqslant \lfloor x_t^{P'} \rfloor + 1\}.$$

au moment de leur création. Par défaut, cette évaluation est donnée par $g(P'_j) := v(P') \leqslant v(P'_j)$ (car $P'_j \subset P'$, mais généralement inégalité stricte car $x^{P'}$ non réalisable pour les fils).

Les fonctions de séparation en base

Ici on supposera que $x^{P'} = x(I, B_-, B_+)$, avec $x^{P'}_{B_-} = \underline{x}_{B_-}$, $x^{P'}_{B_+} = \overline{x}_{B_+}$ à composantes entières, et on envisage de créer les fils ($x^{P'}_t \notin \mathbb{Z}$ astreint d'être entier $\Longrightarrow t \in I$)

$$P'_1 = \{x \in P' : x_t \leqslant \lfloor x^{P'}_t \rfloor\}, \quad P'_2 = \{x \in P' : x_t \geqslant \lfloor x^{P'}_t \rfloor + 1\}.$$

Notons que la base finale du père n'est généralement pas réalisable pour le fils P'_j, et une itération simplex dual sortant t de I donnerait un point de base avec valeur $\widetilde{g}(P'_j) \in [v(P'), v(P'_j)]$.

- "bland": Prendre le premier indice $t \in I$ avec $x^{P'}_t \notin \mathbb{Z}$ astreint d'être entier;
- "blandmod": Comme "bland", sauf que pour la fonction de choix "best" on prend $\widetilde{g}$ au lieu de g;
- "mvp": Trouver $t \in I$ avec $x^{P'}_t$ (astreint d'être entier) le plus éloigné de $\mathbb{Z}$;
- "mvpmod": Comme "mvp", pour "best" on prend $\widetilde{g}$ au lieu de g;
- "peb": Prendre l'indice $t \in I$ avec $x^{P'}_t \notin \mathbb{Z}$ astreint d'être entier maximisant $|v(P'_1) - v(P'_2)|$. Comme le calcul de $|v(P'_1) - v(P'_2)|$ pour tout indice t potentiel est trop coûteux, on prend $|\widetilde{g}(P'_1) - \widetilde{g}(P'_2)|$.

Séparation par pénalités hors base (”phb”)

”peb” prometteur mais trop coûteux. Compromis: ”phb”. Ici on supposera que

$$x^{P'} = x(I, B_-, B_+), \quad \text{avec } x^{P'}_{B_-} = \underline{x}_{B_-}, x^{P'}_{B_+} = \overline{x}_{B_+} \text{ à composantes entières,}$$

et on envisage de créer les fils ($t \in B_- \cup B_+$ avec x_t astreint d'être entier $\Longrightarrow x^{P'}_t \in \mathbb{Z}$)

si $t \in B_-$: $\quad P'_1 = \{x \in P' : x_t = \underline{x}_t\}$ (copie du père), $\quad P'_2 = \{x \in P' : x_t \geqslant \underline{x}_t + 1\}$

si $t \in B_+$: $\quad P'_1 = \{x \in P' : x_t \leqslant \overline{x}_t - 1\}$, $\quad P'_2 = \{x \in P' : x_t = \overline{x}_t\}$ (copie du père)

c'est-à-dire, un des deux fils est une copie du père (sans calcul). Avec variables duales $\underline{w}$, $\overline{w}$ du père

si $t \in B_-$: $\quad v(P'_1) = \widehat{g}(P'_1) := v(P'), \quad v(P'_2) \geqslant \widehat{g}(P'_2) := v(P') + \underline{w}_t \geqslant v(P')$

si $t \in B_+$: $\quad v(P'_1) \geqslant \widehat{g}(P'_1) := v(P') + \overline{w}_t \geqslant v(P'), \quad v(P'_2) = \widehat{g}(P'_2) := v(P').$

Pour pouvoir vite élaguer le fils P'_j avec ”grand” $v(P'_j)$, on choisit l'indice $t \in B_- \cup B_+$ avec x_t astreint d'être entier maximisant $|\widehat{g}(P'_1) - \widehat{g}(P'_2)|$ comme approximation de $|v(P'_1) - v(P'_2)|$.

Autrement, on associe à chaque t candidat une pénalité $\underline{w}_t + \overline{w}^t = |d(I)^t|$ et on choisit celui qui est le plus pénalisé.

Simulations numériques sous Scilab

Lancer Scilab, choisir dans le menue [Fichier] le répertoire `ro/graphs`, et par [Fichier/Exec...] le fichier `my_examples.sci`.
Ce fichier fait appel au fichier `my_simplex.sci` qui comporte

- une procédure `simplex` aux variables doublement bornées (primal ou dual, suivant que l'on donne une base de départ);
- `my_simplex.sci` comporte également une procédure `bb_simplex` récursive mettant en oeuvre l'algorithme du branch-and-bound, ainsi qu'une visualisation de l'arborescence crée. Dans cette arborenscence on note aux sommets la valeur optimale $v(P_j)$ de chaque sous-problème, et aux arcs trois informations: la contrainte supplémentaire permettant de créer ce sous-problème, l'ordre de création de ce sous-problème, et l'ordre de traitement de ce sous-problème;
- une procédure `use_ampl` permettant de faire appel à AMPL avec solveur cplex pour résoudre sous Scilab des problèmes d'optimisation linéaire;
- une procédure `si_init` permettant d'initialiser quelques paramètres (comme des paramètres d'affichage à l'écran), voir `http://math.univ-lille1.fr/~bbecker/simplex/ro_scilab.html`

Exercice 11

On dispose de 4 machines (de capacité en temps d'utilisation fixée) pour fabriquer 10 produits (en quantité donnée). On suppose que si on décide de commencer la fabrication d'un produit *p* sur une machine *m* alors ce produit sera entièrement réalisé par cette machine *m*. On connaît les coûts de fabrication de chaque produit sur chaque machine. Voici les données

	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	capacité
m1	13	9	1	3	14	15	7	8	9	10	9
m2	13	9	20	6	17	7	9	16	17	1	10
m3	18	15	11	14	13	8	8	12	17	10	20
m4	5	4	19	2	6	15	19	4	15	20	3
à produire	3	5	10	1	2	3	5	3	8	2	

Affecter les produits aux machines de manière à minimiser le coût de fabrication. Modéliser sous AMPL (fichier de données `exo11.dat`).

Variante 1: on décide de terminer un exemplaire d'un produit sur la machine sur laquelle on a commencé. Variante 2: tous les exemplaires d'un produit donné sont réalisés sur la même machine (affectation généralisée).

L'utilisation des variables bivalentes dans la modélisation

L'utilisation des variables bivalentes permet de modéliser

- des implications (si création d'un dépôt alors nous imposons nouvelles contraintes) ou équivalences;
- des états logiques (en cas de création de A et/ou B on veut aussi créer C) traduits par une linéarisation;
- des contraintes disjonctives (on souhaite que r parmi n contraintes soient valables, par exemple une parmi deux);
- des fonctions/objectifs affines par morceaux (des remises par pallier);
- des problèmes MaxiMax (maximiser le plus grand parmi un certain nombre d'objectifs);
- des problèmes de couverture, de partitionnement, de découpe et d'affectation.

Conventions: dans ce chapitre, $g(x)$ et $g_j(x)$ désigne des expressions (affines ou non) en des variables x (discrètes ou continues), et δ désigne des variables (parfois vectorielles) bivalentes.

Bien entendu, dans vos modèles sous AMPL on choisit des noms plus parlants.

Variables indicatrices pour inégalités (1)

Pour une expression algébrique $g(x)$, l'inégalité

$$g(x) \leqslant M\delta \quad \text{pour un paramètre } M \in \mathbb{R} \text{ et une variable } \delta \in \{0, 1\}$$

peut signifier une augmentation d'une capacité par M si on décide de construire un entrepôt ($\delta = 1$). On peut également dire qu'elle traduit une implication: si par d'autres contraintes on sait **a priori** que $g(x) \in [-\infty, M]$ pour un $M > 0$ alors

$$\delta = 0 \quad \Longrightarrow \quad g(x) \leqslant 0$$

ou, autrement dit, $g(x) > 0 \Longrightarrow \delta = 1$ (mais nous n'aimons pas des inégalités strictes).
Attention, la contrainte ne traduit pas équivalence (il se peut que $\delta = 1$ et $g(x) \leqslant 0$) mais généralement on n'a pas besoin d'une telle équivalence. De toute façon, si la seule autre occurrence de δ est dans l'objectif, avec un coût unitaire positif, on aura équivalence pour la solution optimale.

Pour une résolution efficace on a intérêt de choisir M le plus petit possible.

Variables indicatrices pour inégalités (2)

Rappel: si on sait **a priori** que $g(x) \in [-\infty, M]$ pour un $M > 0$ alors, avec une variable bivalente δ,

$$\Big(\delta = 0 \quad \Longrightarrow \quad g(x) \leqslant 0\Big) \iff g(x) \leqslant M\delta.$$

Bien entendu, pour l'implication $\delta = 0 \Longrightarrow g(x) \geqslant 17$ on regarde l'expression $17 - g(x)$, et donc notre hypothèse se traduit par une connaissance a priori d'une borne inférieure pour $g(x)$.
Application au voyageur de commerce: on sait a priori que $u_i - u_j + 1 \leqslant n$, donc

$$\Big(\delta_{i,j} = 1 \Longrightarrow u_i - u_j + 1 \leqslant 0\Big) \quad \iff \quad u_i - u_j + 1 \leqslant n(1 - \delta_{i,j}).$$

Variables indicatrices pour égalités

Pour modéliser l'équivalence

$$\delta = 0 \quad \Longleftrightarrow \quad g(x) \leqslant 0$$

on a besoin d'avoir bien plus d'informations a priori sur $g(x)$. Supposons **a priori** que $g(x) \in \{0\} \cup [m, M]$ avec $0 < m < M$. Cela peut se traduire par l'arrêt d'une production, ou la production entre un seuil minimum m et un seuil maximum M. Alors par une table de vérité

$$\Big(\delta = 0 \Longleftrightarrow g(x) = 0\Big) \quad \Longleftrightarrow \quad \left\{ \begin{array}{l} \delta = 0 \Longrightarrow g(x) \leqslant 0 \\ \delta = 1 \Longrightarrow g(x) \leqslant M, -g(x) \leqslant -m \end{array} \right\}$$

$$\Longleftrightarrow \quad \left\{ \begin{array}{l} g(x) \leqslant \delta M \\ -g(x) \leqslant -\delta m \end{array} \right\}$$

Application : fonction à seuil. Si on souhaite exprimer un objectif $f(x) = 0$ si $g(x) = 0$, mais $f(x) = a + bg(x)$ pour $g(x) > 0$ (coût de démarrage et coût de fonctionnement), il suffit d'écrire

$$f(x) = a\delta + bg(x).$$

Expressions logiques avec ”or” et sommes de variables décisionnelles

Avec $\delta_A, \delta_B, ... \in \{0, 1\}$ désignent des variables de décision, la relation $\delta_C = \delta_A \ \texttt{or} \ \delta_B$ est équivalent à

$$\left\{ \begin{array}{l} \delta_C = 0 \Longrightarrow \delta_A + \delta_B \leqslant 0 \\ \delta_C = 1 \Longrightarrow 1 - \delta_A - \delta_B \leqslant 0 \end{array} \right\} \Longleftrightarrow \left\{ \begin{array}{l} \delta_A + \delta_B \leqslant 2\delta_C \\ 1 - \delta_A - \delta_B \leqslant (1 - \delta_C) \end{array} \right\}$$

Parfois, nos exigences sont moindres

- $\delta_A \leqslant \delta_B$ traduit que l'évennement A implique B;
- $\delta_A + \delta_B \geqslant \delta_C$ traduit que la C nécessite aussi A ou B;
- $\delta_A + \delta_B + \delta_C \geqslant 1$ traduit qu'au moins un des trois A, B, C est vrai;
- $\delta_A + \delta_B + \delta_C = 1$ traduit que l'un et un seul de A, B, C est vrai.
- Par conséquent, $B = A_1 \ \texttt{or} \ A_2 \ \texttt{or} \ ... \ \texttt{or} \ A_n$ peut aussi être traduit par l'ensemble des inégalités $\delta_{A_1} + \delta_{A_2} + ... + \delta_{A_n} \geqslant \delta_B$, et pour $j = 1, 2, ..., n : \delta_{A_j} \leqslant \delta_B$.

Une combinaison de ces idées: ”si on fait A_1 ou A_2 alors on souhaite aussi faire B_1, B_2 ou B_3” se traduit par

$$\begin{aligned} &\left(\delta_{A_1} + \delta_{A_2} \geqslant 1 \Longrightarrow \delta = 1 \Longrightarrow \delta_{B_1} + \delta_{B_2} + \delta_{B_3} \geqslant 1\right) \\ \Longleftrightarrow &\left(\delta_{A_1} + \delta_{A_2} \leqslant 2\delta, 1 - \delta_{B_1} - \delta_{B_2} - \delta_{B_3} \leqslant (1 - \delta)\right). \end{aligned}$$

Expressions logiques ”and”, ”xor”, et linéarisation

La traduction $\delta_C = \delta_A \delta_B$ de $\delta_C = \delta_A$ `and` δ_B nécessite une linéarisation:

$$\delta_C = \delta_A \delta_B \iff \left\{ \begin{array}{l} \delta_C = 0 \Longrightarrow \delta_A + \delta_B - 1 \leqslant 0 \\ \delta_C = 1 \Longrightarrow 2 - \delta_A - \delta_B \leqslant 0 \end{array} \right\} \iff \left\{ \begin{array}{l} \delta_A + \delta_B - 1 \leqslant \delta_C \\ 2 - \delta_A - \delta_B \leqslant 2(1 - \delta_C) \end{array} \right.$$

De même, on peut linéariser par substitution $y = \delta x$ le produit entre une variable décisionnelle et une variable continue : si on sait a priori que $x \in [0, M]$, alors

$$y = \delta\, x \iff \left\{ \begin{array}{l} 0 \leqslant y \leqslant x \\ \delta = 0 \Longrightarrow y \leqslant 0 \\ \delta = 1 \Longrightarrow x - y \leqslant 0 \end{array} \right\} \iff \left\{ \begin{array}{l} 0 \leqslant y \leqslant x \\ y \leqslant \delta M \\ x - y \leqslant M(1 - \delta). \end{array} \right\}$$

Pour exprimer le ”ou” exclusif $\delta_C = \delta_A$ `xor` δ_B, on peut s’inspirer du ”et”

$$\delta_C = \delta_A + \delta_B - 2\,\delta_A\,\delta_B \iff \left\{ \begin{array}{l} \delta_C = \delta_A + \delta_B - 2\,\delta \\ \delta_A + \delta_B - 1 \leqslant \delta \\ 2 - \delta_A - \delta_B \leqslant 2(1 - \delta) \end{array} \right\}$$

Contraintes disjonctives (1)

Problème: On souhaite qu'au moins r parmi les n contraintes $g_j(x) \leqslant 0$ pour $j = 1, ..., n$ soient valables, sans savoir a priori lesquelles.

Exemple : on fixe le début des tâches dans la construction d'une maison sachant leur durées, et un carreleur ne peut pas s'occuper simultanément de la salle de bain et de la cuisine.

Solution: il suffit d'introduire des variables indicatrices $\delta_j \in \{0, 1\}$ avec

$$\text{pour } j = 1, 2, ..., n : \quad \delta_j = 1 \Longrightarrow g_j(x) \leqslant 0$$

(autrement, $g_j(x) \leqslant M_j(1 - \delta_j)$ avec des bornes a priori $g_j(x) \leqslant M_j$), et de s'assurer que $\delta_1 + \delta_2 + ... + \delta_n \geqslant r$.

Contraintes disjonctives (2)

Problème: On souhaite que exactement $r = 1$ parmi les $n = 2$ contraintes $g_1(x) \leqslant 0$ et $g_2(x) \leqslant 0$ soit valable, sans savoir a priori laquelle.

Si pour d'autres raisons on sait que les contraintes $g_1(x) \leqslant 0, g_2(x) \leqslant 0$ s'excluent mutuellement (voire notre carreleur), il suffit d'introduire une variable bivalente $\delta_1 = 1 - \delta_2$.

Exemple : On sait a priori qu'une expression $g(x)$ ne prend que des valeurs dans $[1 - M, M - 1] \cap \mathbb{Z}$. Comment imposer que $g(x) \neq 0$ ($g(x) \leqslant -1$ ou $g(x) \geqslant 1$)?

$$g(x) + 1 \leqslant (1 - \delta)M, \quad 1 - g(x) \leqslant \delta M, \quad \delta \in \{0, 1\}.$$

Exercice 12

On souhaite colorier la carte des régions en France (on exclut la Corse et les territoires outre-mer) avec un nombre minimum de couleurs sachant que deux régions ayant une frontière commune devraient avoir une couleur différente pour pouvoir les distinguer.

Dans le tableau à droite on donne des abréviations des régions, ainsi que le nom de leurs voisins respectifs. L'idée derrière la modélisation est que 4 couleurs devraient être suffisantes pour colorier un graphe planaire.

Région	Régions limitrophes
N	Pi,
Pi	N,HN,Ch,IF
HN	Pi,BN,Ce,IF
BN	HN,Br,PL,Ce,
Br	BN,PL,
PL	Br,BN,Ce,PC
Ce	PL,BN,HN,IF,Bo,Au,Li,PC
IF	Ce,Ch,Bo,Pi,HN,
Bo	Ce,RA,Au,FC,Ch,IF,
Au	Ce,LI,Bo,RA,LR
Li	Ce,PC,Aq,MP,Au
PC	Ce,PL,Aq,
Aq	PC,Li,MP
MP	Aq,Li,LR,
RA	Au,PCA,Bo,FC
LR	Au,MP,PCA,LA
PCA	LR,RA
LA	LR
FC	RA,Al,Lo,Ch,Bo
Al	FC,Lo,
Lo	FC,Al,Ch,
Ch	FC,Lo,Bo,IF,Pi,

Modélisation sous AMPL (voir fichier de données `exo12.dat`)

Exercice 13

On se donne un ensemble de n usines, chacune peut soit fournir une production du même bien comprise entre $\underline{P}_i$ et $\overline{P}_i$, avec un coût de démarrage et un coût par unité produite, soit être à l'arrêt (sans engendrer des coûts).
On souhaite satisfaire une demande $C = 700$ au moindre coût.
Voici les données numériques

Usine	production minimum	production maximum	coût de démarrage	coût unitaire de production
U1	40	120	120	32
U2	120	230	230	28
U3	80	240	960	20
U4	70	220	890	21
U5	80	220	880	26

Modélisation sous AMPL (voir fichier de données `exo13.dat`)

Fonctions affines par morceaux (les remises)

Problème: On se donne $\lambda_0 < \lambda_1 < ... < \lambda_n$ et une fonction continue définie sur $[\lambda_0, \lambda_n]$ par la formule $f(x) = f_j(x)$ si $x \in [\lambda_{j-1}, \lambda_j]$, $j = 1, ..., n$. Comment calculer $f = f(x)$ pour un $x \in [\lambda_0, \lambda_n]$ donné?
Solution: on introduit des variables indicatrices $\delta_1, ..., \delta_n$ avec

$$\Big(\delta_j = 1 \implies x \in [\lambda_{j-1}, \lambda_j]\Big) \iff \left\{ \begin{array}{l} \lambda_{j-1} - x \leqslant (1 - \delta_j)(\lambda_{j-1} - \lambda_0) \\ x - \lambda_j \leqslant (1 - \delta_j)(\lambda_n - \lambda_j) \end{array} \right\}$$

Pour prendre en compte que x appartient à un et un seul de ces intervalles, on imposera $\delta_1 + ... + \delta_n = 1$, et on peut en déduire que

$$f = \delta_1 f_1(x) + ... + \delta_n f_n(x)$$

(attention, comme f est supposée d'être continue, les cas limites $x = \lambda_j$ ne posent pas de problèmes). Comme x est une variable, il convient de linéariser $y_j = \delta_j f_j(x)$. Si a priori $0 \leqslant f_j(x) \leqslant M_j$ pour tout $x \in [\lambda_0, \lambda_n]$ alors

$$y_j = \delta_j g_j(x) \iff \left\{ \begin{array}{l} 0 \leqslant y_j \leqslant f_j(x) \\ y_j \leqslant \delta_j M_j \\ f_j(x) - y_j \leqslant M_j(1 - \delta_j) \end{array} \right\}$$

MiniMax versus MaxiMax

Etant donné des expressions affines $g_1(x), ..., g_m(x)$, dans un problème MiniMax on cherche à résoudre

$$\min_x \max_j g_j(x).$$

Par introduction d'une variable artificielle $z \in \mathbb{R}$ on peut facilement se ramener à un programme linéaire

$$\min\{z : \forall j = 1, ..., m,\ g_j(x) \leqslant z\}.$$

Par contre, en utilisant la même approche pour un problème MaxiMax

$$\max_x \max_j g_j(x)$$

on cherche à maximiser z sous les contraintes disjonctives

$$g_1(x) \geqslant z \ \text{ ou } \ g_2(x) \geqslant z \ \text{ ou } \ ... \ \text{ ou } \ g_m(x) \geqslant z.$$

Quelques problèmes type

Problème de couverture

On se donne un ensemble S à n éléments, et $\Sigma = \{\sigma_1, ..., \sigma_m\}$ un ensemble de sous-ensembles $\sigma_j \subset S$, et pour chaque σ_j un coût c_j. Le problème de couverture consiste à trouver au moindre coût un sous-ensemble Σ_0 de Σ de sorte que la réunion des éléments de Σ_0 donne bien l'ensemble S entier.
La modélisation se fait à l'aide de m variables décisionnelles ($\delta_j = 1$ signifie que l'on sélectionne σ_j pour Σ_0), permettant d'exprimer l'objectif comme $\min(c_1\delta_1 + c_2\delta_2 + ... + c_m\delta_m)$. Pour tester la couverture, on dispose d'une table de vérité pour l'appartenance

$$\forall s \in S : \quad \sum_{j=1}^{m} \chi(s,j)\delta_j \geqslant 1, \qquad \chi(s,j) := \begin{cases} 1 & \text{si } s \in \sigma_j, \\ 0 & \text{si } s \notin \sigma_j. \end{cases}$$

En remplaçant $\geqslant 1$ par $\geqslant 2$ dans les contraintes, on pourrait par exemple demander que chaque élément soit couvert au moins 2 fois.
Problème de partitionnement: On remplace $\geqslant 1$ par $= 1$ dans les contraintes, car on souhaite trouver une partition Σ_0 de S, c'est-à-dire, tout élément de S doit se retrouver une et une seule fois dans un élément de Σ_0.

Problèmes de couplage

Pour le problème de couplage au coût minimum, les ensembles candidats $\Sigma \subset S \times S$, sont des couples de S, mais cette fois on cherche à marier une personne pas plus qu'une fois

$$\forall s \in S : \quad \sum_{j=1}^{m} \chi(s,j)\delta_j \leqslant 1$$

avec le même objectif que pour le problème de couverture. Par contre, pour le problème de couplage maximum on cherche juste à former un maximum de couples.
A cause de la forme particulière de $\Sigma \subset S \times S$, il existe des algorithmes de résolution de complexité $\mathcal{O}(mn^2)$, basés sur la théorie des graphes.

Application : composition d'équipages d'avions (1)

Tiré de ROADEF : Le bulletin 3(1999),
http://www.roadef.org/content/roadef/bulletins/bulletinNo3.pdf
"PLANIFICATION D'EQUIPAGES", par Jean-François Puget Directeur R&D Optimisation, ILOG.

[...] Nous nous intéresserons plus particulièrement à l'affectation d'équipages. Il s'agit de déterminer les équipages nécessaires pour assurer un certain nombre de vols d'avion donnés. Le problème est habituellement résolu en deux phases. La première phase consiste à réunir les vols en des ensembles cohérents (rotations) pouvant être effectués par un équipage. [...] La deuxième étape est un problème de partitionnement : il s'agit de sélectionner un ensemble de rotations de façon à couvrir chaque vol par une rotation et une seule. Cette deuxième phase peut se modéliser avec une variable de décision binaire par rotation et une contrainte par vol. Pour chaque vol, exactement une rotation incluant ce vol doit être sélectionnée. Autrement dit, la somme des variables correspondant aux rotations incluant ce vol doit être égale à un.

Application : composition d'équipages d'avions (2)

Le modèle résultant est linéaire et peut être résolu, en principe, par les techniques de programmation linéaire en nombres entiers (PLNE). En pratique les modèles sont trop larges pour être résolus directement, car le nombre de colonnes (variables) peut être énorme. Dans ce cas, une technique par génération de colonne (ou décomposition de Dantzig-Wolfe) sera utilisée. Le problème est d'abord résolu en ne considérant qu'un sous ensemble des rotations possibles. Ensuite, on cherche à déterminer si la prise en compte de nouvelles rotations permet d'améliorer le résultat. La théorie de la dualité nous dit que seules les variables (donc les rotations) ayant un coût réduit négatif doivent être prises en compte. Toute la difficulté consiste à ne pas produire toutes les rotations possibles, mais celles dont le coût réduit sera le plus négatif possible.[...]

Problème de découpe

Étant donnés des barres de longueur fixée $L_1, ..., L_r$ (ici une seule longueur L), un fournisseur doit satisfaire un carnet de m commandes, la commande k consistant en b_k morceaux de longueur ℓ_k. On cherchera à satisfaire le carnet en réalisant "au mieux" des découpes dans des barres stockés. Pour cela, on pourra former des formats de découpe, par exemple le format j pour une barre de longueur L sera crée par la détermination d'un vecteur colonne d'entiers $\vec{n} = \vec{n}(j) = (n_1, ..., n_m)^T$, avec $n_k \geqslant 0$ le nombre de bâtons de longueur ℓ_k que l'on programme de faire suivant ce format de découpe, avec chute

$$c^j = L - \lambda\vec{n} \geqslant 0, \qquad \lambda = (\ell_1, ..., \ell_m).$$

En notant ces différents formats de découpe dans une matrice A à m lignes, et les chutes correspondantes dans le vecteur ligne c, notre fournisseur doit résoudre

$$\min\{cx : Ax = b, \forall j : x_j \geqslant 0, x_j \in \mathbb{Z}\}, \quad b = (b_k)_{k=1,...,m},$$

$$\text{où} \quad cx = Lfx - \lambda Ax = Lfx - \lambda b, \quad f = (1, ..., 1).$$

On utilisera donc l'objectif f comptant le nombre de barres utilisés.

Découpes et la génération de colonnes (1)

Malheureusement, comme sur les transparents précédents, le nombre de formats est tellement important qu'il est impossible de déterminer A a priori. Heureusement, pour la forme révisée de Simplex on en a pas besoin.

Oublions les contraintes d'intégrité pour x ce qui nous permet d'appliquer Simplex. Ici on a seulement besoin d'accéder aux matrices A et f entières dans le test d'optimalité (calcul coûts réduits)

$$s = \arg\min_j d(I)^j, \quad d(I)^j = f^j - \mu A^j, \quad \mu = f^I (A^I)^{-1}.$$

La recherche de l'indice s pour une longeur L de barre donnée consiste alors de trouver le format $\vec{n}$ de découpe d'indice j minimisant la chute modifiée $f^j - \mu A^j = 1 - \mu\vec{n}$, c'est-à-dire, on doit résoudre

$$v := \min\{1 - \mu\vec{n} : \lambda\vec{n} \leqslant L, \vec{n} \in \mathbb{Z}^m, \vec{n} \geqslant 0\},$$

un problème de type sac-à-dos (pour chaque longueur de barre $L = L_p$, ici une seule).

Découpes et la génération de colonnes (2)

$$v := \min\{1 - \mu\vec{n} : \lambda\vec{n} \leqslant L, \vec{n} \in \mathbb{Z}^m, \vec{n} \geqslant 0\},$$

Si $v \geqslant 0$ (CSO), on a trouvé la solution optimale, et sinon on connaît $A^s = \vec{n}$ et $f^s = 1$. Donc on peut résoudre notre problème initial (en fractionnaire) sans jamais avoir connu la matrice entière A.

Approche 1 : En partant de $A^I = U \in \mathbb{R}^{m\times m}$, on stocke seulement A^I, il faut donc résoudre un problème de sac-à-dos à chaque itération de Simplex.

Approche 2 : En partant de $A^J = U \in \mathbb{R}^{m\times m}$, on résout par Simplex $\max\{f^J x_J : A^J x_J = b, x_J \geqslant 0\}$, et on ajoute si nécessaire la nouvelle colonne $A^J \leftarrow (A^J, A^s)$.

Des idées similaires sont utilisées pour des problèmes de découpe de tissus (en 2D). La génération de colonnes est aussi l'idée clef dans la décomposition de Dantzig/Wolfe.

```
# Fichier cut.mod
# Originalite de l'implementation AMPL :
# ce fichier comporte deux problemes a resoudre

problem Trouver_decoupes;
# resoudre probleme pour une liste A de formats
param nb_FORMATS integer >= 0, default 0;
param longeur_barre;
set FORMATS := 1..nb_FORMATS;
set BATONS; # nom d'un baton=longueur
param demande_batons {BATONS} >0; #batons a produire
param A{BATONS,FORMATS} integer>=0; #matrice formats
var Decoupes{FORMATS}integer >=0; #nb formats retenus
minimize nb_barres: sum {j in FORMATS} Decoupes[j];
subj to satisfaire_demande {i in BATONS}:
 sum{j in FORMATS} A[i,j]*Decoupes[j]=demande_batons[i];

problem Trouver_format;
# trouver un nouveau format de decoupe (mvp)
param price {BATONS} default 0;
var nb_batons {BATONS} integer >= 0;
minimize Cout_reduit:
 1 - sum {i in BATONS} price[i] * nb_batons[i];
subj to Limite_longeur :
 sum {i in BATONS} i * nb_batons[i] <= longeur_barre;
```

```
reset; option solver cplex;
option solution_round 6;        # arrondi
option solver_msg 0;            # affichage resolution
option show_stats 0;            # affichage simplification
model ../monampl/data/cut.mod;
data ../monampl/data/cut.dat;
problem Trouver_decoupes;
  option relax_integrality 1;  # sans integrite
  option presolve 0;           # sans simplification
problem Trouver_format;
  option relax_integrality 0; option presolve 1;
let nb_FORMATS := 0; # initialiser A
for {i in BATONS} {
  let nb_FORMATS := nb_FORMATS + 1;
  let A[i,nb_FORMATS] := floor (longeur_barre/i);
  let {i2 in BATONS: i2 <> i} A[i2,nb_FORMATS] := 0;
  };
repeat {
  solve Trouver_decoupes;
  let{i in BATONS}price[i]:=satisfaire_demande[i].dual;
  solve Trouver_format;
  if Cout_reduit < -0.00001 then {
     let nb_FORMATS := nb_FORMATS + 1;
     let {i in BATONS} A[i,nb_FORMATS] := nb_batons[i];
     }
  else break;
  }; # ensuite affichage decoupes et solution Simplex
```

Affectation simple

Affecter n machines à n tâches, sachant que la machine i effectuant la tâche j représente un coût c_{ij}. Il s'agit d'un cas particulier d'un problème de transport (une unité par usine=tâche et par client=usine, matrice totalement unimodulaire).

Cette formulation comme problème de transport est également possible si on dispose d'un nombre m de machines différent de n, et si certains machines ne savent que traiter un nombre limité de tâches (lien avec problème de couplage à poids maximum).

Affectation généralisée

$$\begin{aligned}
\text{maximiser} \quad & \sum_{j\in J}\sum_{k\in K} c_{jk}x_{jk} \\
\text{sous les contraintes} \quad & \forall k \in K : \quad \sum_{j\in J} a_{jk}x_{jk} \leqslant b_k \\
& \forall j \in J : \quad \sum_{k\in K} x_{jk} = 1 \\
& \forall j \in J \forall k \in K : \quad x_{jk} \in \{0,1\}.
\end{aligned}$$

On peut par exemple s'imaginer un ensemble de vols $j \in J$ qui doivent tous être effectués une et une seule fois, en utilisant un parmi l'ensemble des avions $k \in K$, sachant que la durée a_{jk} du vol j dépend de l'avion k. Si tout $a_{jk} = 1$ alors on se retrouve avec un problème de transport, mais dans le cas général la matrice n'est plus totalement unimodulaire.

Affectation quadratique

Il s'agit d'un problème d'affectation de n usines à n villes, mais le choix de deux couples (j_1, k_1) et (j_2, k_2) induit un coût (par exemple de communication) qui dépend à la fois de la distance entre deux villes et du couple d'usines en question. On se ramène alors à un objectif de la forme

$$\sum_{j_1,k_1,j_2,k_2=1}^{n} c_{j_1,k_1,j_2,k_2} x_{j_1,k_1} x_{j_2,k_2},$$

c'est-à-dire, minimiser une forme quadratique sous contraintes linéaires, mais la difficulté provient du fait que l'on doit minimiser dans $\{0,1\}^{n^2}$ et pas dans $\mathbb{R}^{n^2}$.

Exercice 14

On se donne dans `exo14.dat` un tableau de distances entre 8 villes en France, ainsi qu'un ensemble d'avions, avec pour chaque avion sa vitesse moyenne (en km/h), et sa disponibilité (en h). L'objectif est d'affecter au moindre coût un des avions à chaque vol (=couple de villes distincts aller et retour), sachant que à la durée du vol il s'ajoute un délai fixe pour préparer l'avion au décalage (qu'il faut ajouter au temps de réservation d'un avion), et sachant que le coût unitaire d'un vol est donné par la distance, auquel il s'ajoute un coût fixe de démarrage.

Dans un deuxième temps, tenir compte du fait qu'un avion arrivant dans une ville x ne peut pas ensuite partir d'une ville différente de x. L'endroit de départ d'un avion n'a pas d'importance, mais il faudra que l'avion revient au point de départ.

Modélisation sous AMPL (voir fichier de données `exo14.dat`)

Exercice 15

Dans un garage automobile, on dispose d'un certain nombre d'examens (diagnostics) $e_1, ..., e_n$ pour détecter des pannes $p_1, p_2, ..., p_m$. Un examen est soit positif soit négatif (par exemple tester si le moteur marche encore), mais la raison peut être multiple. Pour chaque examen j, on se donne dans un tableau son coût c_j, et la liste des pannes que cet examen peut détecter (sachant que le moteur ne dit rien sur l'état des pneus): $\chi(j,k) = 1$ si l'examen j peut détecter la panne k, et 0 sinon.

examen\panne	p1	p2	p3	p4	p5	c_i
e1	1	0	1	1	0	110
e2	0	1	1	1	0	120
e3	1	1	0	0	1	30
e4	0	0	1	1	1	40
e5	1	1	0	1	0	50

Trouver le sous-ensemble des examens de moindre coût permettant de trouver une panne quelconque à coup sûr. *Indication:* Notons par D_k l'ensemble des examens pouvant détecter une panne k donnée. Préciser la nature de $D_{k,\ell} = (D_k \setminus D_\ell) \cup (D_\ell \setminus D_k)$ pour un couple de pannes (k, ℓ). Montrer que l'on peut se ramener à un problème de couverture de coût minimum. (`exo15.dat`)

Exercice 16 : Problème d'ordonnancement

Nous souhaitons fabriquer un certain nombre de pièces (ici a, b, et c) en les faisant passer sur un certain nombre de machines (ici P, Q, R, S), le temps d'usinage d'une pièce sur une machine étant connu. Une machine ne peut pas usiner simultanément plusieurs pièces, et, en plus, l'ordre de passage sur les machines est connu pour chaque pièce

$$a : P \to R \to Q, \quad b : P \to Q \to R \to S, \quad c : P \to Q \to R \to S,$$

mais des temps d'attente entre machines sont permis.
Trouver l'ordre de passage qui permet de terminer l'ensemble des pièces le plus vite possible.

Modélisation sous AMPL (voir fichier de données `exo16.dat`)

Exercice 17

Une équipe de basket espagnole est composée de 11 joueurs dont 4 ailiers, 3 joueurs de base et 4 joueurs pivot. Une équipe comporte au plus deux joueurs hors CEE, et au moins 4 joueurs de nationalité espagnole.

Vous êtes demandé de composer sur une période de trois semaines l'équipe parmi un ensemble de 41 joueurs. D'une semaine à une autre, il faudra garder au moins huit joueurs. Pour chaque joueur et chaque semaine on connaît les gains obtenus (en euros) par match s'il fait partie de l'équipe.

AMPL (voir fichier `exo17.dat`)

Nom	sem0	rôle	origine	sem1	sem2	se
Urtasun	x	ailier	espagnol	86552		
Rubio	x	base	espagnol	79397		
Trias	x	pivot	espagnol	67669	77819	61
Salenga	x	ailier		59265		
Gabini	x	ailier		48074		
Fernandez	x	base	espagnol	14079		
Gasol	x	pivot	espagnol			
Price	x	base	horsCEE			
Santiago	x	pivot	horsCEE			
Pietrus	x	pivot				
De la Fuente	x	ailier	espagnol			
Rancik		pivot		172467	154751	
Clancy		pivot	horsCEE	97601		
Recker		ailier	horsCEE	75000		
Weis		pivot		63516		
Varda		pivot		53351	6778	
Keselj		ailier		50052	43266	
Fajardo		pivot	espagnol	49749	57211	39
Alzamora		pivot	espagnol	36000		
Lopez		base	espagnol	34747	36005	
Brown		base		33445		
Savovic		ailier	espagnol	32725		
Baldo		pivot		30113	34630	14
Douglas		ailier	horsCEE	28609		
Bueno		pivot	espagnol	27145		
Cazorla		ailier	espagnol	25659	21678	
Guardia		pivot	espagnol	24189		
Longin		ailier		23878		
Chiacig		pivot		23800		
Beechum		ailier		21945		
Dragic		base		20393	23452	26
Vazquez		ailier	espagnol	18900		
Zengotitabengoa		ailier	espagnol	18198	20927	24
Garcia		pivot	espagnol	16806	19327	22
Sonko		base		16646		
Santamaria		base	espagnol	14613	16805	6
Doblas		pivot	espagnol	12329		
Oliver		base	espagnol	6067		
Norris		base		5123		
Archibald		pivot		3586		
Martin		pivot	espagnol	1488		