TD programmation robuste

Exercice1. Programme robuste pour le plus court chemin

Soit un graphe orienté G=(V,E) , les arcs *e* munis de poids c_e. s, t∈V sommet source et sommet puit.

1° Donner un programme mathématique (linéaire) pour calculer le plus court chemin de la source au puit.

Maintenant, le graphe a un ensemble S de scenarios possibles sur les valeurs des arcs : c_e^S s=1,…,p.

2° Donner le programme mathématique qui calcule le chemin qui minimise le pire coût des scenarios.

3° Donner le programme mathématique qui calcule le chemin qui minimise la moyenne ordonnée de la valeur des chemins de la source au puit.

Correction

1° Le programme pour trouver le plus court chemin de s à t.

Variables d'arcs $x_{ij} = \begin{cases} 1 \text{ si on prend l'arc } (i,j) \\ 0 \text{ sinon} \end{cases}$

$$\min \sum_{e\in E} c_e x_e \text{ s.c} \begin{cases} \sum_{(j,i)\in E} x_{ji} - \sum_{(i,j)\in E} x_{ij} = \begin{cases} -1 \; si \; i = s \\ 1 \; si \; i = t \\ 0 \; si \; i \neq s \; et \; t \end{cases} & (1) \\ x_{ij} \in \{0,1\} \text{ pour tout } (i,j) \in E & (2) \end{cases}$$

Maintenant, plusieurs scenarios *s* sont possibles pour les coûts des arcs : c_e^S s=1,…,p.

2° Programme robuste : Minimiser le coût du pire scenario. Se prémunir contre le pire scenario.

$$\min_{\gamma,x} \gamma \text{ s.c} \begin{cases} \gamma \geq \sum_{e\in E} c_e^s x_e \;\; s = 1, \dots, p \\ (1), (2) \end{cases}$$

3° Moyenne OWA

Soit une suite $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p \geq 0$ avec $\sum_{i=1}^{p} \lambda_i = 1$.

Minimiser la moyenne ordonnée des coûts du pire scenario vers le moins pire. Ici, il faut se référer au modèle décrit dans le cours.

$$\min_{\alpha,\beta,x} \sum_{s=1}^{p} \alpha_s + \beta_s$$

$$\text{s.c.} \begin{cases} \alpha_s + \beta_{s'} \geq \lambda_{s'} \sum_{e\in E} c_e^s x_e \;\; s, s' = 1, \dots, p \\ (1), (2) \end{cases}$$

Exercice2. Sac-à-dos robuste

On considère un problème de sac-à-dos en variables binaires.

(P) $\max_x \sum_{i=1}^n c_i x_i$ s.c. $\begin{cases} \sum_{i=1}^n a_i x_i \leq b \quad (sac - \text{à} - dos) \\ x \in \{0,1\}^n \end{cases}$

Les poids a_i des objets sont incertains. La valeur nominale de a_i est $\bar{a}_i > 0$ et l'amplitude de variation autour de la valeur nominale est $\hat{a}_i > 0$. Les 3 valeurs possibles pour chaque poids sont $a_i = \begin{cases} \bar{a}_i - \hat{a}_i \\ \bar{a}_i \\ \bar{a}_i + \hat{a}_i \end{cases}$

On veut une solution qui maximise la valeur du sac et robuste par rapport à l'incertitude. On suppose qu'au plus Γ poids vont dévier de leur valeur nominale.

1° Donner (P_x) le programme linéaire en nombres entiers qui donne la protection nécessaire que l'on doit ajouter à la contrainte de sac-à-dos de telle sorte qu'une solution *x* soit toujours réalisable. Montrer que l'on peut relâcher les conditions d'intégrité dans (P_x).

2° (P_x) étant un programme linéaire, dualiser (P_x) et donner la version robuste et linéaire du programme (P).

3° Donner les formules de récurrence permettant de résoudre (P_x).

4° A partir de ces formules de récurrence, proposer une alternative au programme robuste précédent.

5° Mêmes questions en supposant maintenant que ce sont les utilités c_i des objets qui sont incertaines.

Correction

1° La protection de la contrainte de sac-à-dos est donnée par :

$\max_{\Delta} \sum_{i=1}^n \hat{a}_i \Delta_i x_i$ s.c. $\Delta_i \in \{-1,0,1\}$ pour $i = 1, \dots, n$ et $\sum_{i=1}^n |\Delta_i| \leq \Gamma$ (C)

La variable $\Delta_i = -1$ si a_i passe en-dessous de sa valeur nominale $a_i = \bar{a}_i - \hat{a}_i$

La variable $\Delta_i = +1$ si a_i passe au-dessus de sa valeur nominale $a_i = \bar{a}_i + \hat{a}_i$

Mais les variables $x_i \geq 0$ donc on prendra toujours $\Delta_i \geq 0$. D'où le programme en 0-1 :

(P_x) $\max_{\Delta} \sum_{i=1}^n \hat{a}_i \Delta_i x_i$ s.c. $\Delta_i \in \{0,1\}$ $i = 1, \dots, n$ et $\sum_{i=1}^n \Delta_i \leq \Gamma$ (C)

Mais le problème étant linéaire en les variables Δ_i et les coefficients de Δ_i dans la contrainte de cardinalité (C) étant 1, on peut relâcher les contraintes d'intégrité sur Δ_i. On obtient finalement le programme linéaire suivant :

(LP_x) $\max_{\Delta} \sum_{i=1}^n \hat{a}_i \Delta_i x_i$ s.c. $\Delta_i \in [0,1]$ $i = 1, \dots, n$ et $\sum_{i=1}^n \Delta_i \leq \Gamma$ (C)

2° Le dual de (LPx) est $\min_{\mu,\lambda} \mu\Gamma + \sum_{i=1}^n \lambda_i \ \ s.c. \ \begin{cases} \mu + \lambda_i \geq \hat{a}_i x_i \ \ i = 1, \dots, n \quad (1) \\ \mu \geq 0, \lambda_i \geq 0 \ \ i = 1, \dots, n \quad (2) \end{cases}$

D'où le programme robuste :

$$\max_{x,\mu,\lambda} \sum_{i=1}^{n} c_i x_i \text{ s.c. } \begin{cases} \sum_{i=1}^{n} \bar{a}_i x_i + \mu\Gamma + \sum_{i=1}^{n} \lambda_i \leq b \quad (sac - \text{à} - dos) \\ (1), (2) \\ x \in \{0,1\}^n \end{cases}$$

On peut retirer le min du dual puisque pour maximiser l'objectif $\sum_{i=1}^{n} c_i x_i$, les variables μ, λ vont minimiser le terme ajouté sur la contrainte de sac-à-dos afin de moins contraindre.

3° On note $A_{i\gamma}$ = la protection pour les variables de 1 à i lorsque γ poids dévient de leur valeurs nominales.

Les formules de récurrence sont :

$$\begin{cases} A_{i,0} = 0 \; i = 1, \dots, n \\ A_{1,1} = \hat{a}_1 x_1 \, , A_{1,2} = \hat{a}_1 x_1 \\ A_{i,\gamma} = \max\{A_{i-1,\gamma} \, , A_{i-1,\gamma-1} + \hat{a}_i x_i\} \; i = 2, \dots, n \; et \; \gamma = 1, \dots, \min\{i, \Gamma\} \end{cases}$$

4° D'où le programme robuste :

$$\max_{x,A} \sum_{i=1}^{n} c_i x_i \text{ s.c. } \begin{cases} \sum_{i=1}^{n} \bar{a}_i x_i + A_{n\Gamma} \leq b \quad (sac - \text{à} - dos) \\ A_{i,0} = 0 \qquad i = 1, \dots, n \\ A_{1,1} = \hat{a}_1 x_1 \quad , \; A_{1,2} = \hat{a}_1 x_1 \\ A_{i,\gamma} \geq A_{i-1,\gamma} \qquad i = 2, \dots, n \; et \; \gamma = 1, \dots, \min\{i, \Gamma\} \; (3) \\ A_{i,\gamma} \geq A_{i-1,\gamma-1} + \hat{a}_i x_i \; i = 2, \dots, n \; et \; \gamma = 1, \dots, \min\{i, \Gamma\} \; (4) \\ x \in \{0,1\}^n \end{cases}$$

Là encore les variables *A* vont se positionner au plus bas pour moins contraindre le problème afin de maximiser l'objectif et on aura bien $A_{i,\gamma}$ saturant l'une des 2 contraintes (3) ou (4) ce qui donnera bien le max présent dans les relations de récurrence.

5° Cette fois, les variations sont sur les coefficients de la fonction objectif c_i. La valeur nominale de c_i est $\bar{c}_i > 0$ et l'amplitude de variation autour de la valeur nominale est $\hat{c}_i > 0$.

Dans l'objectif, pour un *x* fixé les coefficients c_i vont se mettre au plus bas afin de pénaliser l'objectif. D'où le terme de pénalité qu'il faudra retrancher à l'objectif :

$$\max_{\Delta} \sum_{i=1}^{n} \hat{c}_i \Delta_i x_i \text{ s.c. } \Delta_i \in \{0,1\} \; i = 1, \dots, n \text{ et } \sum_{i=1}^{n} \Delta_i \leq \Gamma \; (\mathrm{C})$$

Le programme robuste est :

$$\max_{x} \sum_{i=1}^{n} \bar{c}_i x_i \; - \max_{\Delta} \sum_{i=1}^{n} \hat{c}_i \Delta_i x_i \text{ s.c. } \begin{cases} \sum_{i=1}^{n} a_i x_i \leq b \quad (sac - \text{à} - dos) \\ \Delta_i \in \{0,1\} \; i = 1, \dots, n \text{ et } \sum_{i=1}^{n} \Delta_i \leq \Gamma \; (\mathrm{C}) \\ x \in \{0,1\}^n \end{cases}$$

Il faut positionner les variables x de façon à maximiser le premier terme sans que le second ne devienne trop <0. La solution optimale nous donne la valeur optimale que l'on obtiendra dans le pire des cas. On ne pourra pas descendre en dessous de cette valeur quelque-soit le scenario se produisant.

Pour les mêmes raisons que précédemment, on peut relâcher les conditions d'intégrité sur les variables Δ_i et on peut dualiser la pénalité.

On obtient :

$$\max_{x,\mu,\lambda} \sum_{i=1}^{n} \bar{c}_i x_i \; - (\mu\Gamma + \sum_{i=1}^{n} \lambda_i) \text{ s.c. } \begin{cases} \sum_{i=1}^{n} a_i x_i \leq b \quad (sac - \text{à} - dos) \\ \mu + \lambda_i \geq \hat{c}_i x_i \;\; i = 1, \dots, n \quad (1) \\ \mu \geq 0, \lambda_i \geq 0 \;\; i = 1, \dots, n \quad (2) \\ x \in \{0,1\}^n \end{cases}$$

Pour l'approche par programmation dynamique, on note $C_{i\gamma}$ = la pénalité pour les variables de 1 à i lorsque γ utilités dévient de leur valeurs nominales. On obtient le programme robuste :

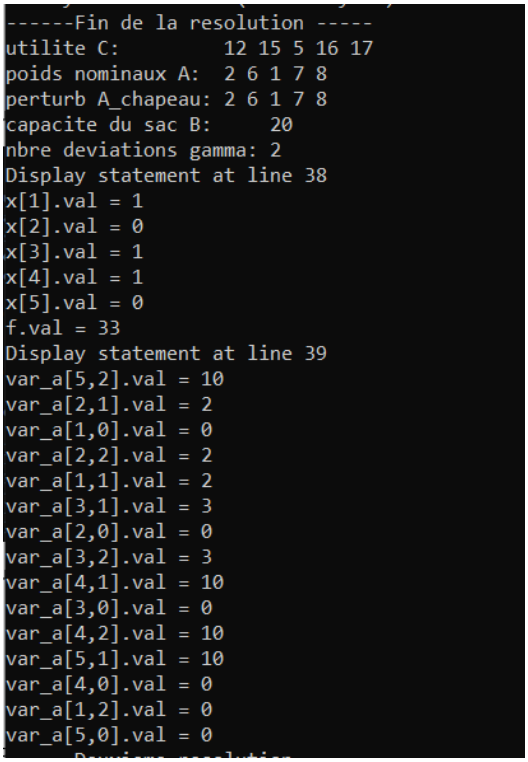
$$\max_{x,C} \sum_{i=1}^{n} \bar{c}_i x_i \; - C_{n\Gamma} \text{ s.c. } \begin{cases} \sum_{i=1}^{n} a_i x_i \leq b \quad (sac - \text{à} - dos) \\ C_{i,0} = 0 \qquad\qquad i = 1, \dots, n \\ C_{1,1} = \hat{c}_1 x_1 \;\; , \;\; C_{1,2} = \hat{c}_1 x_1 \\ C_{i,\gamma} \geq C_{i-1,\gamma} \qquad i = 2, \dots, n \; et \; \gamma = 1, \dots, \min\{i, \Gamma\} \quad (3) \\ C_{i,\gamma} \geq C_{i-1,\gamma-1} + \hat{c}_i x_i \; i = 2, \dots, n \; et \; \gamma = 1, \dots, \min\{i, \Gamma\} \quad (4) \\ x \in \{0,1\}^n \end{cases}$$

Résultats numériques pour incertitude sur la contrainte, modèle dualité question 2°

```
------Fin de la resolution -----
utilite C:          12 15 5 16 17
poids nominaux A:   2 6 1 7 8
perturb A_chapeau: 2 6 1 7 8
capacite du sac B:      20
nbre deviations gamma: 2
Display statement at line 35
x[1].val = 1
x[2].val = 0
x[3].val = 1
x[4].val = 1
x[5].val = 0
f.val = 33
Display statement at line 36
mu.val = 1
lambda[1].val = 2
lambda[2].val = 0
lambda[3].val = 0
lambda[4].val = 6
lambda[5].val = 0
```

On remarque que la protection de la contrainte vaut 10= $\mu\Gamma + \sum_{i=1}^{5} \lambda_i$. Alors que 9 suffirait car les 3 variables $x_1 = x_3 = x_4$ sont à 1 et Γ=2 donc au plus 2 poids varient et on arrive au maximum à 7+2. Mais il s'avère que mettre une protection à 9 au lieu de 10 c'est-à-dire assouplir la contrainte de sac-à-dos d'une unité, ne change rien à la valeur optimale de l'objectif qui reste à 33.

Résultats numériques pour incertitude sur la contrainte, modèle prog. dynamique question 4°



Idem la protection vaut 10 : $A_{5,2} = 10$. Alors que 9 suffirait mais mettre la protection à 9 c'est-à-dire la contrainte de sac-à-dos assouplie d'une unité, ne change pas la valeur de l'objectif qui reste à 33.

Exercice 3. Programmation robuste avec recours : localisation d'entrepôt sans contraintes de capacité

Soient n entrepôts, m clients. Chaque client doit être affecté à un unique entrepôt. Un client ne peut être affecté à un entrepôt que si ce dernier est ouvert. Le coût d'ouverture d'un entrepôt j est c_j. Le coût de raccordement d'un client i à un entrepôt j est q_{ij}. On doit ouvrir des entrepôts (parmi les n disponibles) de telle sorte que les coûts d'ouverture et de raccordement des clients aux entrepôts est minimum. Le problème se modélise par un programme linéaire ($P_{0\text{-}1}$) en variables 0-1 :

variables x_j=1 si entrepôt j est ouvert, =0 sinon ; y_{ij}=1 si le client i est raccordé à l'entrepôt j, =0 sinon.

($P_{0\text{-}1}$) $$\min_{x,y} \sum_{j=1}^{n} c_j x_j + \sum_{i=1}^{m} \sum_{j=1}^{n} q_{ij} y_{ij}$$

sous les contraintes $$\begin{cases} \sum_{j=1}^{n} y_{ij} = 1 & i = 1, \dots, m \\ \sum_{i=1}^{m} y_{ij} \leq m x_j & j = 1, \dots, n \\ x_j, y_{ij} \in \{0,1\} & i = 1, \dots, m; j = 1, \dots, n \end{cases}$$

La première famille de contraintes sont les contraintes d'affectation d'un client i (i=1 à m), à exactement un des entrepôts.

La deuxième famille impose que l'on ne peut raccorder aucun client à un entrepôt j fermé (x_j=0).

On remarque que dans la solution optimale chaque client i est raccordé au dépôt le plus proche ouvert c'est-à-dire y_{ij^*} = 1 où j*=argmin{q_{ij} : j=1,…, n tel que x_j=1 }. Reste à trouver quels entrepôts ouvrir.

Les coûts de raccordement des clients aux entrepôts ne sont en fait pas connus avec certitude. Ils dépendront d'infrastructures futures. On a prévu p scénarios de coûts de raccordement. On doit décider maintenant quels entrepôts construire. Les raccordements des clients aux entrepôts sont des décisions qui seront prises plus tard. On veut trouver la solution minimisant les coûts de construction des entrepôts plus le pire coût de raccordement des clients qui pourra arriver. On appelle cette solution *solution optimale robuste*.

On considère l'instance suivante : n=3, m=5,

Coût construction	Entrepôt 1	Entrepôt 2	Entrepôt 3
	1	4	1

Il y a p=2 scenarios de coûts de raccordement. Le premier scenario est donné ci-dessous :

Coût raccordement Scenario 1	Client 1	Client 2	Client 3	Client 4	Client 5
Entrepôt 1	1	3	8	1	1
Entrepôt 2	7	4	2	2	5
Entrepôt 3	2	5	7	4	4

Le deuxième scénario de coût de raccordement est donné dans le tableau ci-dessous :

Coût raccordement scenario 2	Client 1	Client 2	Client 3	Client 4	Client 5
Entrepôt 1	4	3	2	4	2
Entrepôt 2	2	6	1	8	1
Entrepôt 3	12	7	5	9	8

Q1- Dans le cas de l'instance, trouver la solution optimale robuste. Pour cela remplir le tableau suivant :

Entrepôts construits	Coût construction	Coût min de raccordement des clients dans scenario 1	Coût min de raccordement des clients dans scenario 2	Pire coût de raccordement des clients	Coût total	Solution optimale robuste (mettre une x)
1						
2						
3						
1, 2						
1, 3						
2, 3						
1,2,3						

Q2- Modéliser le problème par un programme linéaire en variables 0-1 (n+mnp variables 0-1, 1 variable réelle).

Correction

Q1- Dans le cas de l'instance, trouver la solution optimale robuste.

Entrepôts construits	Coût construction	Coût min de raccordement des clients dans scenario 1	Coût min de raccordement des clients dans scenario 2	Pire coût de raccordement des clients	Coût total	Solution optimale robuste (mettre une x)
1	1	14	15	15	16	X
2	4	20	18	20	24	
3	1	22	41	41	42	
1, 2	5	8	11	11	16	X
1, 3	2	13	15	15	17	
2, 3	5	14	18	18	23	
1,2,3	6	8	11	11	17	

Q2- Modéliser le problème par un programme linéaire en variables 0-1 (n+mnp variables 0-1, 1 variable réelle).

On repart du modèle sans scenario donné dans l'énoncé. On indexe les variables d'affectation des clients *i* aux entrepôts *j* par un numéro de scenario *s* : y_{ij}^s.

$y_{ij}^s = 1$ si le client i est raccordé au dépôt j dans le scenario s et 0 sinon.

Et ensuite, on minimise la fonction incluant les coûts de construction des entrepôts et le max sur les scenarios des coûts de raccordement des clients aux entrepôts.

($P_{Robuste\text{-}0\text{-}1}$) $\min_{x,y} \sum_{j=1}^{n} c_j x_j + \gamma$

sous les contraintes
$$\begin{cases} \sum_{j=1}^{n} y_{ij}^s = 1 \quad i = 1, \dots, m, s = 1, \dots, p \\ \sum_{s=1}^{p} \sum_{i=1}^{m} y_{ij}^s \leq pmx_j \qquad j = 1, \dots, n \\ \gamma \geq \sum_{i=1}^{m} \sum_{j=1}^{n} q_{ij}^s y_{ij}^s \qquad s = 1, \dots, p \\ x_j, y_{ij}^s \in \{0,1\} \quad i = 1, \dots, m; j = 1, \dots, n; s = 1, \dots, p \end{cases}$$

Programme en Mathprog (GLPK)

```
# localisation entrepots robuste avec variables de recours
param n; # nombre entrepots
param m; # nombre clients
```

```
param nb_scen; # nombre de scenarios
param c{1..n}; # couts construction entrepots
param dist{1..m,1..n,1..nb_scen}; # distances des clients aux entrepots
#
var x{1..n} binary; # var. d' ouverture entrepot
var y{1..m,1..n,1..nb_scen}>=0; # var. rattachement client aux entrepots
var gamma;
#
minimize z:sum{j in 1..n}c[j]*x[j]+gamma;
subject to
contr_affectation{i in 1..m,s in 1..nb_scen}: sum{j in 1..n}y[i,j,s]=1;
contr_ouverture{j in 1..n}: sum{i in 1..m,s in 1..nb_scen}y[i,j,s]<=m*nb_scen*x[j];
# contraintes pour modeliser le pb de recours
contr_recours{s in 1..nb_scen}: gamma>=sum{i in 1..m, j in
1..n}dist[i,j,s]*y[i,j,s];
```