DAUPHINE UNIVERSITÉ PARIS

L3 INFO
PROGRAMMATION LINÉAIRE
TRAVAUX PRATIQUES

TP2 : RÉSOLUTION GÉNÉRALE D'UN PROGRAMME LINÉAIRE AVEC AMPL

1 Résoudre avec AMPL un programme linéaire en le décrivant différemment

Si le Programme Linéaire est beaucoup plus gros au niveau du nombre de variables et de contraintes, il existe une façon plus concise de le mettre au format AMPL. Cela consiste à définir dans un premier temps, donc premier fichier, le modèle général, puis ensuite préciser, dans un second fichier, les données particulières pour lesquelles on souhaite la résolution. Le modèle général est alors écrit sous forme algébrique. Ainsi, AMPL peut résoudre un modèle avec différentes données.

1.1 Modèle d'un PL

Le modèle générique de notre PL peut s'exprimer sous la forme suivante :

Étant donné :	P un ensemble de produits
	R un ensemble de ressources
	a_{ij} la quantité de ressources i de R nécessaire pour une unité de chaque produit j de P
	b_i la quantité de ressource i disponible pour chaque i de R
	c_j le prix de vente d'une unité de produit j, pour tout j de P
Définir les variables :	$X_j =$ nombre de produits j à fabriquer, pour chaque j de P
Maximiser :	$\sum_{j \in P} c_j \, X_j$
Sous les contraintes :	$\sum_{j \in P} a_{ij} \, X_j \leq b_i$ pour tout i de R
	$X_j \geq 0$ pour tout j de P

Ce modèle décrit en fait une infinité de PL. Si nous donnons des valeurs spécifiques aux données, le modèle devient un problème spécifique. Notre exemple du cours, n'en est alors qu'une instance particulière.

Pour rentrer ce modèle sous AMPL, il existe des instructions qui permettent de spécifier chacune de ses composantes, à savoir : les ensembles, les paramètres, les variables, l'objectif et les contraintes. Le modèle de production générique, créé dans un fichier dont l'extension sera **.mod**, est donc avec AMPL le suivant :

```
set  P;
set R;
param b{i in R};
param c{P};
param a{R,P};
var X{j in P};
maximize CA : sum {j in P} c[j]*X[j];
subject to contrainte {i in R} : sum {j in P} a[i,j]*X[j] <= b[i];
s.t. positivite {j in P}: X[j] >=0;
```

La syntaxe est la suivante :

- le mot-clé `set` permet de définir une collection finie d'éléments. Ces éléments peuvent être numériques ou non et sont souvent utilisés comme des indices. En particulier un ensemble particulièrement utilisé est de type $1 \dots n$ où n est à valeur entière précisée avant, et dans ce cas l ?ensemble est celui de tous les entiers allant de 1 à n,
- le mot-clé `param` permet de déclarer un paramètre qui devra être spécifié,
- `b{i in R}` (qui est équivalent à `b{R}`) permet de traduire "il y a un b_i pour chaque i de R" et pour accéder à b_i il faut écrire `b[i]`,
- `a{R,P}` permet de traduire "il y a un a_{ij} pour chaque i élément de R et chaque j élément de P" et pour accéder à a_{ij} il faut écrire $a[i, j]$,
- l'instruction `sum{j in P}` permet de faire une somme sur les indices j variant dans l'ensemble P,
- l'instruction `s.t.` est équivalente à **subject to**. Si le nom de la contrainte est suivie d'accolades, c'est que l'on définit une collection de contraintes. Les contraintes de bornes supérieures, inférieures, positivité peuvent être faites au moment de la déclaration des variables en mettant `var X{j in P} >=0 <=M;`

Les ensembles, paramètres et variables doivent être déclarés avant d'être utilisés. Les noms utilisés pour identifier les objets du modèle (ensembles, paramètres, variables, contraintes, fonction objectif) doivent être uniques.

1.2 L'instanciation d'un Programme Linéaire via ses données

Considérons le programme linéaire suivant :

$$(PL) \begin{cases} \max & 2000\, x_1 & + & 3000\, x_2 & & & \\ s.c. & & & & & & \\ & x_1 & + & 6\, x_2 & \leq & 30 & (R1) \\ & 2\, x_1 & + & 2\, x_2 & \leq & 15 & (R2) \\ & 4\, x_1 & + & x_2 & \leq & 24 & (R3) \\ & x_1, & & x_2 & \geq & 0 & \end{cases}$$

Pour représenter ce PL considérée comme une instance du modèle, il faut préciser un fichier de données qui lui correspond, dont l'extension sera **.dat**. Le fichier de données contiendra les valeurs des ensembles et des paramètres du modèle. L'ordre est arbitraire : il n'est pas nécessairement le même que celui du modèle. Les données de notre problème sont du type :

```
set P := prod1, prod2;
set R := R1, R2, R3;
param b:=
R1 30
R2 15
R3 24;
param c :=
prod1 2000 prod2 3000;
param a:
prod1 prod2 :=
R1 1 6
R2 2 2
R3 4 1;
```

1.3 Résolution d'une instance d'un modèle

Il faut effectuer cette fois les instructions suivantes :

```
ampl: model fichier.mod;
ampl: data fichier.dat;
ampl: solve;
```

1.4 Amélioration du modèle

On peut donner une version dite améliorée de ce modèle en effectuant les transformations suivantes :

- donner des noms plus explicites aux ensembles, paramètres, variables. Par exemple l'ensemble P peut être nommé `PRODUIT`, l'ensemble R `RESSOURCE`, le paramètre b `dispo`, le paramètre c `prix`, les variables X `PRODUCTION`,
- en mettant les contraintes de positivité lors des déclarations des variables,
- des commentaires, commençant avec `#`, peuvent être ajoutés pour expliquer le modèle.

Exercice 1 :

1. Écrire dans le fichier *exemple2.mod* le modèle générique du (PL) ci-dessus en appliquant les règles présentées ci-dessus.
2. Écrire dans le fichier *exemple2.dat* les données du (PL) ci-dessus pour le modèle correspondant
3. Résoudre ce PL et vérifier votre résultat avec celui obtenu au TP1.

2 Prise en compte de modifications

Une fois le PL généré, il est facile de prendre en compte un certain nombre de modifications du PL. Supposons que :

- l'on ajoute un nouveau produit qui serait fabriqué par l'usine : alors il suffit de modifier le fichier de données en ajoutant un nom de produit supplémentaire et d'ajouter ses caractéristiques dans les différents paramètres.

- l'on ajoute une borne supérieure pour chaque produit : alors il suffit d'ajouter dans le fichier du modèle un nouveau paramètre, représentant cette borne supérieure, existant pour chaque variable et le définir comme borne lors de la déclaration des dites variables. Il faut également ajouter dans le fichier de données les valeurs de ce paramètre pour chacune des variables.

Exercice 2 :

1. On souhaite à présent résoudre (PL) ci-dessus mais avec un produit supplémentaire nommé *produit 3* qui se vend 2500 euros et nécessite 2 heures de main d'oeuvre, 2 unités de R2 et 1 unités de R3.

   Modifier ce qu'il faut dans *exemple2.mod* et/ou *exemple2.dat* pour résoudre optimalement ce nouveau PL.
2. Afficher la valeur optimale de chaque variable.
3. Si le nombre d'unités vendues quotidiennement est au maximum pour *le produit 1* de 4, pour *le produit 2* de 2 et pour le *produit 3* de 3, que faut-il modifier pour obtenir la résolution optimale de ce nouveau PL ?
4. Afficher les nouvelles valeurs optimales de chaque variable et la fonction objectif dans un fichier solution.

Exercice 3 :

On suppose que sont données dans le tableau ci-dessous, pour 100 g de chaque type d'aliment, le nombre de calories, les protéines (en grammes), le calcium (en mg) et la quantité (en I.U.) de vitamines A qu'il contient. De plus, on a le coût d'achat de 100 g de chacun de ces aliments.

	pain	viande	pommes de terre	chou	lait	gélatine
calories	1254	1457	318	46	309	1725
protéines	39	73	8	4	16	43
calcium	418	41	42	141	536	0
vitamine A	0	0	70	860	720	0
coût pour 100g	0.3	1	0.05	0.08	0.23	0.48

On sait que quotidiennement il faut au moins consommer 3000 calories, 70 g de protéines, 800 mg de calcium et 500 I.U. de vitamines A. On cherche la quantité de chacun des aliments qu'il faut consommer quotidiennement pour satisfaire les besoins minimum en calories, protéines, calcium et vitamine A tout en minimisant le coût total.

1. Donner le modèle associé à ce problème en le commentant et utilisant des noms de paramètres, contraintes, variables explicites. Le sauver dans le fichier *alimentation.mod*.
2. Donner le fichier de données, nommé *alimentation.dat* associé à l'instance que l'on souhaite résoudre.
3. Quelle est la solution optimale ? Quelles sont les contraintes saturées ? Donner la quantité de chaque aliment qu'il faut consommer et le nombre de calories, protéines, calcium et vitamines A associés.

Exercice 4 :

Résoudre des exercices de TD de votre choix avec le logiciel AMPL.